National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

Digital Image Warping

by

Sydney Lee ©

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Science

Department of Computing Science

Edmonton, Alberta
Fall 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-95067-6

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Sydney Lee**

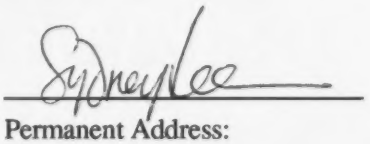
TITLE OF THESIS: Digital Image Warping

DEGREE: **Master of Science**

YEAR THIS DEGREE GRANTED: 1994

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Permanent Address:
1B Regent Court
7 Li Kwan Avenue
Hong Kong

Date: June 7th 1994

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled Digital Image Warping submitted by **Sydney Lee** in partial fulfillment of the requirements for the degree of **Master of Science**.

Supervisor: Dr. A. Basu

Examiner: Dr. B. Joe

External: Dr. R.E. Rink

Date: June 3rd 1994

To my family and friends

Abstract

A new warping technique is presented that maps images of arbitrary planar shapes onto each other. The algorithm is called the Radial Transform which is based on an extension of digital-straight line representation that resamples the images in the polar coordinate system. The Radial Transform described uses boundary maps to compute the forward mapping functions. The technique not only produces superior image quality than existing algorithms, but also simplifies the correspondence problem and the resampling process. In addition, the technique also transforms image shapes that are *not topologically equivalent*, unlike several other methods discussed in the literature.

This thesis also presents another warping algorithm which is a combination of the Radial Transform and a 3-pass algorithm — an existing technique for arbitrary image mapping. This technique is called the Radial-Axes Transform that warps 2D images in three 1D transformations. The technique provides more local control over the Radial Transform.

Acknowledgment

I would like to thank my supervisor Dr. A. Basu for his inspiration of this work, his encouragement, support, and patience throughout my graduate studies. Special thanks to my examination committee Dr. B. Joe and Dr. R.E. Rink for their time and effort.

Thanks to Carol Smith for her help on X-Windows programming, Edith Drummond and Britta Nielsen for their general advice.

I am grateful to my instructor Brendan Skelton for his support throughout my studies.

Appreciation is expressed to my friends Linda Xie, Mein Woo, Kacheong Yeung, Collins Chien, Vikas Nehru, and Chu-Kiat Lim for their encouragement, discussions, and valuable suggestions, and to all the good folks at the University of Alberta.

Contents

1 Introduction 1

1.1 Spatial Transformation 2

1.1.1 Forward Mapping 3

1.1.2 Inverse Mapping 3

1.2 Image Resampling 4

1.2.1 Image Reconstruction 4

1.3 Antialiasing 5

1.3.1 Filtering 5

1.3.2 Sampling 5

1.4 Thesis Roadmap 6

2 Background 7

2.1 Rigid Warping Algorithms 7

2.2 Free-form Warping Algorithms 8

3 Problem Description 9

3.1 Arbitrary Shapes 10

4 Arbitrary-Shapes Warping 12

4.1 Skeleton-based Warping 12

4.1.1 3-pass Transformation 13

4.1.2 Establishing Correspondences 16

4.2 Radial Transform 17

4.2.1 Spatial Transformation 17

4.2.1.1 Image Origin 18

4.2.1.2 Correspondence Problem 19

4.2.1.3 Sampling Radial Lines 20

4.2.1.4 Bresenham Line Algorithm 20

4.2.1.5 Elimination of "holes" 21

4.2.1.6 Radial-Line Correspondences 25

4.2.2 Image Resampling 25
4.2.2.1 Bilinear Interpolation 26
4.2.2.2 Linear Interpolation 27
4.2.2.3 Fant's Resampling Algorithm 27
4.2.3 Inverse Spatial Transformation 29
4.3 Mapping Between Radial Lines 33
4.4 Theoretical Analysis 36
4.4.1 Image Quality 36
4.4.2 Speed Performance 43
4.4.3 Memory Requirement 44
4.5 Radial-Axes Warping 44
4.5.1 Spatial Transformation 44
4.5.1.1 Image Origin 45
4.5.2 3-pass Transformation 46
4.5.3 Inverse Spatial Transformation 47
5 Performance Comparisons 48
5.1 Signal-to-noise Ratio Comparisons 48
5.2 Time Comparisons 52
6 Radial Morphing 56
7 Conclusion 59
8 Gallery 61
Bibliography 66

List of Tables

4.1 Input pixels sampled for the Skeleton-based and the Radial methods 41
4.2 Cost of performing all steps of Radial warping 43
4.3 Memory costs 44
5.1 CPU-time range. 53
5.2 Cost of performing Radial warping. 53
5.3 Cost of performing Skeleton-based warping 54
5.4 CPU-time range for same image size. 55

List of Figures

3.1 Arbitrary image shapes. 9
3.2 Mapping between rectangular images 10
3.3 Mapping between arbitrary image shapes. 11
4.1 Conceptual Skeleton-based method. 13
4.2 Skeleton-based transformation 14
4.3 An example of intermediate images transformed by the Skeleton-based method 15
4.4 Conceptual Radial transform 17
4.5 Image origins and sampling directions for Radial method 18
4.6 Bounding-box traversal 20
4.7 Real line L and digital line l on Cartesian coordinates 21
4.8 Real lines in real space 22
4.9 Three adjacent radial-digital lines that pass through the origin 23
4.10 Linear interpolation for a digital line. 27
4.11 Fant's 1D resampling algorithm (redrawn) [Fant86] 28
4.12 Radial transformation. 30
4.13 An example of the intermediate images transformed by the Radial transform. 32
4.14 Interpolation error around image boundary. 32
4.15 The first example for mapping of a radial line between two arbitrary topological different image shapes. 33
4.16 Mapping input image pixels from two line segments to one for Figure 4.15 34

4.17 The second example for mapping of a radial line between two arbitrary topological different image shapes. 35
4.18 Mapping input image pixels from two line segments for Figure 4.17 35
4.19 "Even" image stretching. 36
4.20a Enlarge image S to D with radial lines and sampled pixels 37
4.20b Superimposed images S and D with radial lines and sampled pixels 38
4.21a Sampled pixels for Skeleton-based method. 39
4.21b Pixels after resampling along the u-axis. 39
4.21c Pixels after resampling along v-axis. 40
4.22 Errors in sampling for Skeleton-based method. 40
4.23 Image division for Radial-Axes method. 45
5.1a Images with high frequencies in the frequency domain 49
5.1b Images with low frequencies in the frequency domain. 49
5.2 Mapping functions used for performance comparisons 50
5.3 Signal-to-noise ratio tests 51
8.1 Ozone 61
8.2 The Earth is square? 62
8.3 Downcast 62
8.4 Vortex 63
8.5 Beaver 63
8.6 Silence 64
8.7 Alien 65
8.8 Eggle 65

Chapter 1

Introduction

Image warping is a 2D geometric transformation that modifies the spatial relationships between pixels in an image. Imagine an image printed on a sheet of rubber and then stretching or squeezing this sheet according to some predefined rules: geometric transformations are often described as rubber sheet transformations. Image warping plays an important role in both image processing and computer graphics.

In image processing, transformations often are defined over the 2D coordinate system. However, it is generally not possible to formulate a single set of analytic expressions that describes the geometric distortion process over the entire image. To overcome this difficulty, four-corner mapping is often used to warp quadrilaterals onto quadrilaterals. In computer graphics, warping amounts to reparameterization of coordinate systems [Wolberg90]. The 2D source image is mapped onto 3D objects followed by a projection onto the 2D screen. This process is known as texture mapping [Heckbert86]. Most previous work in both the above research areas has concentrated on transformations that can be formulated by analytic expressions. However, there is a lack of attention in the literature to the mapping between arbitrary image shapes such as hand-drawn closed curves. The rectilinear Cartesian coordinates are not suitable for representing arbitrary image shapes [Wolberg88][Wolberg89]. These arbitrary image shapes cannot be easily defined mathematically in general, and hence they may not be represented by 2D transformations. They are also not suitable for four-corner mapping. Moreover, they are not parameterized.

We present two simple algorithms that map an arbitrary image onto any other arbitrary image. The first algorithm is called Radial transform in which the user defines an origin in the image. The source image is converted from Cartesian coordinates to polar coordinates to produce an intermediate image; then the intermediate image is warped to the destination image. A possible application of the Radial transform is discussed in Chapter 6. The second algorithm is called Radial-axes transform which is similar to the first one, except that images are divided into regions before the warp takes place. First, the images are divided into four regions by two axes specified by the user. Next, each region in the source is mapped to the corresponding destination using a 3-pass technique. Both algorithms are based on an extension of digital-straight line representation that resamples the images in the polar coordinate system. In addition to scaling, the algorithms allow lateral transformation. Unlike previous methods discussed in the literature, we used boundary maps to specify objects' boundaries in order to permit the mapping of arbitrary shapes even if they are not topological equivalents. The following sections describe the three main components of image warping: spatial transformations, resampling, and antialiasing.

1.1 Spatial Transformation

A geometric transformation maps one coordinate system onto another by means of a mapping function. The mapping function is known as the spatial transformation that establishes a sampling grid which defines the spatial relationship between all points in the input and output images.

The spatial transformation mapping function may be specified by analytic expressions, four-corner mapping, or control-grid mapping [Wolberg90]. In this thesis, we do not consider spatial transformation that are represented by analytic expressions.

In general, the mapping function can take two forms: forward mapping and inverse mapping.

1.1.1 Forward Mapping

In forward mapping, input pixels are scanned in scanline order, but the results are free to leave in any order. Each input pixel (x_s, y_s) is mapped onto the output at positions (x_d, y_d) specified by the mapping functions X and Y.

$$(x_d, y_d) = (X(x_s, y_s), Y(x_s, y_s)) \tag{1.1}$$

Since there is no guarantee that X and Y will generate integer values, the input pixels are mapped from the integer coordinate values onto the real-valued positions. Since the input and output coordinates are different in nature (one is discrete and the other is continuous), it is not advisable to implement the spatial transformation as a point-to-point mapping or holes and overlaps may occur. Holes occur when the input pixels are mapped to sparse positions on the output coordinate. Overlaps occur when several input samples are mapped to one output pixel.

The spatial transformation is usually implemented by a four-corner mapping where the input pixels are treated as square patches that may be mapped into quadrilaterals in the output image. Since the input can lie anywhere in the output image, input pixels often straddle several output pixels, or lie embedded in one. Therefore, it is necessary to calculate a weighted contribution of each input pixel that covers an output pixel. Costly intersection tests are needed to compute the weights and the coverage. To avoid this problem, the input can be sampled adaptively based on the size of the projected quadrilateral.

1.1.2 Inverse Mapping

In inverse mapping, output pixels are generated in scanline order. Each output position is projected into the input image by the mapping functions X^{-1} and Y^{-1}, which are the inverse mapping function of X and Y respectively.

$$(x_s, y_s) = (X^{-1}(x_d, y_d), Y^{-1}(x_d, y_d)) \tag{1.2}$$

In contrast with forward mapping, the output pixels are projected from the integer lattice onto the input at real-valued positions. The value of the sampled point is copied onto the output pixel. However, the input pixels are often defined on an integer lattice, therefore an interpolation stage is required to sample the input values at the undefined real-valued positions. The details of this interpolation stage are discussed in the Image Resampling Section.

Inverse mapping assumes that all output pixels are computed but not all input pixels are mapped. Therefore, artifacts may arise if large amounts of input pixels are discarded while calculating the output. Inverse mapping is the most common method used because it generates the output in scanline order, and therefore it guarantees each output pixel is mapped.

1.2 Image Resampling

Image resampling refers to image reconstruction followed by sampling. After establishing an inverse spatial transformation, the input image must be reconstructed or interpolated for sampling. Sampling theory is essential to digital image transformations. In the discrete domain, undesirable artifacts can arise as a result of geometric transformation, therefore sampling theory is central to the study of sampled-data systems.

1.2.1 Image Reconstruction

Both input and output digital images are restricted to lie on integer coordinates. In inverse mapping, the output pixels are passed through a mapping function that generates a sampling grid which is used to resample the input. Unfortunately, each point in the new resampling grid may take on any continuous value. Therefore, it is necessary to interpolate a continuous surface through the discrete input samples. This process is known as image reconstruction. After image reconstruction, the interpolated continuous surface can be sampled at any position to yield an output value.

The quality of the output image depends on the accuracy of the interpolated image which in turn depends on the interpolation function. Well known interpolation

algorithms include cubic convolution, bilinear, nearest neighbor, cubic spline and convolution with a sinc function. For speed, efficiency and acceptable image quality, we used bilinear interpolation for our algorithms.

1.3 Antialiasing

Antialiasing is the process of filtering the high frequencies of a signal that cause aliasing. Aliasing occurs when a signal is undersampled and hence the high frequency components overlap causing undesired artifacts. The problem can be solved by bandlimiting the signal or increasing the sampling rate. The first solution uses low-pass filters to bandlimit the signal to conform to the Nyquist rate. Bandlimiting the signal to levels below the Nyquist rate effectively cuts off the offending high frequencies. The second solution is to increase the sampling rate, thereby placing the replicated spectra farther apart — this effectively separates the overlapping high frequencies that cause aliasing. This method is superior in terms of image quality, but it is costly and may be difficult to achieve. In practice, both methods are used together to combat aliasing. The following sections briefly discuss the basic ideas behind the two methods.

1.3.1 Filtering

A bandlimited signal can be generated by passing the original signal through a low-pass filter before sampling. Since this process must be performed before the sampling, it is known as prefiltering. Any low-pass filters may be used to bandlimit a signal. Basically, there are two types of filters: space-invariant filter and space-variant filter. In space-invariant filter, the kernel is constant as the filter scans the entire image. In space-variant filter, the kernel varies with position.

1.3.2 Sampling

There are two classes of sampling algorithms that may be used to combat aliasing: regular sampling algorithms and irregular sampling algorithms. The former techniques use regular sampling grids to collect the image samples where as the latter techniques use irregular sampling grids to sample the input data.

1.4 Thesis Roadmap

The road map of the thesis is laid out as follows: Chapter 2 provides the background work. The definition of the problem and the goal of the thesis are presented in Chapter 3. Chapter 4 describes an existing algorithm and introduces a new algorithm. The performance of these algorithms are analyzed. The experimental results of the performance are presented in Chapter 5. Chapter 6 describes a possible application of this algorithm. The conclusion is given in Chapter 7. A selection of example pictures is shown in Chapter 8.

Chapter 2

Background

In this study, we dichotomized warping techniques into rigid warping and free-form warping algorithms. In rigid warping algorithms, the same set of transformation functions is applied to all points in the image. Transformations such as rotation, scaling, and perspective mapping are rigid warping algorithms. Since they are frequently used in Computer Graphics and Computer Vision, we first describe some techniques that optimize these transformations. In free-form warping algorithms, the transformation functions apply to only a subset of the image points, that is, not all image points are mapped in the same way. These algorithms will be discussed next. However, with all the rules there are exceptions. Some algorithms can be categorized as rigid or free-form warping algorithms as we shall see that in Chapter 3. For mapping between arbitrary shapes, only free-form warping algorithms should be used. The reason becomes clear when we describe the problem in Chapter 3.

2.1 Rigid Warping Algorithms

In 1979, Braccini and Marino showed that transformations such as rotation and scale can be implemented by sampling a texture map in scanline order and placing the results along the path of a Bresenham digital line [Braccini and Marino79]. "Holes" that may appear between adjacent lines are filled with extra pixels, which results in some redundancy. Nevertheless, the algorithm achieves high speed-up for image rotations.

Catmull and Smith propose a general 2-pass forward mapping scanline algorithm that transforms a 2D image using two 1D transformations [Catmull and Smith80]. The 2-pass algorithm transforms the rows of the source image completely before transforming the columns. The 1D transformation simplifies the resampling process, and it is computationally less expensive than traditional transformations that operate entirely in 2D. Moreover, it is suitable for hardware implementation.

Based on Catmull and Smith's 2-pass algorithm, Wolberg and Boult propose a general forward transformation that uses spatial look-up tables to specify any mapping functions [Wolberg and Boult89]. The spatial look-up tables avoid the computation of the inverse function required by the 2-pass algorithm[Catmull and Smith80].

2.2 **Free-form Warping Algorithms**

Fiume et al. propose a conformal texture mapping where images are represented by convex polygons [Fiume et al.87]. They use the Schwarz-Christoffel transformation to construct the conformal maps. Schwarz-Christoffel mapping can be costly to compute and extremely complicated analytically.

The first algorithm that maps between arbitrary shapes is proposed by Wolberg. Since this is the only algorithm that maps between arbitrary shapes, we will discuss it in detail later. Wolberg introduces a skeleton-based image warping technique that employs a thinning (or erosion) process on the images, similar to peeling the skins off an onion [Wolberg88][Wolberg89]. This process creates a "thinning path". The sampled (or extracted) image is then reparameterized from Cartesian coordinates into a new (u, v) space in which u runs along the image boundary, and v runs radially inward. A 3-pass algorithm is used to map the source image to the destination image. The final result is placed back in Cartesian coordinates using the thinning paths.

Chapter 3

Problem Description

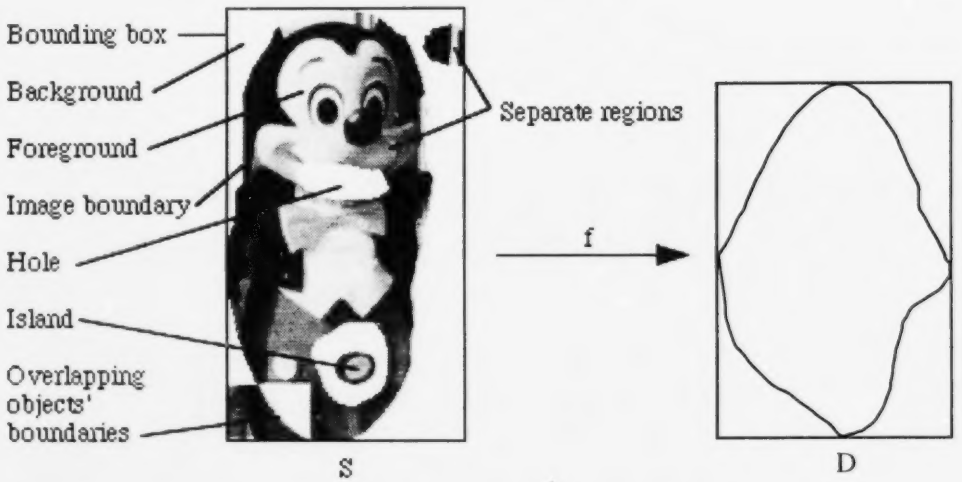


Figure 3.1: Arbitrary image shapes.

We would like to map an arbitrary source image, S, onto any arbitrary destination region, D, using a mapping function f (see Figure 3.1). In fact, both S and D are sub-images that can be extracted by tracing objects' boundaries with a pointer on the screen. Each sub-image is extracted together with its minimum bounding box, with the edges of the bounding box parallel to the Cartesian coordinate axes. Pixels that are within image boundaries are designated as foreground and the remaining ones as background. Unlike Wolberg's technique, our algorithms do not put any restrictions on the topology of the source and destination regions. For example, Figure 3.1 shows that S contains

regions of hole, island, overlapping objects' boundaries, and two separate regions. Moreover, our algorithms also allow scaling and lateral mapping.

3.1 **Arbitrary Shapes**

This section reviews the difficulties encountered when mapping between arbitrary shapes. The main difficulty is finding correspondences in both images.

A function, f, that maps S onto D requires that both images be parameterized such that correspondences may be established. Consider a rectangular image on the Cartesian Coordinate (see Figure 3.2). Image S is bounded by its four edges. The position of each interior point can be determined relative to those edges or the four corner points. Since image D also has four corner points, we can use these points as the corresponding points between S and D. Consider f as a simple scaling mapping function. The correspondences a_s, b_s, c_s and d_s are mapped to a_d, b_d, c_d and d_d respectively. Since the positions of the four correspondences are known in both S and D, then the position of each interior point can be determined, hence all points in S can be mapped onto D.

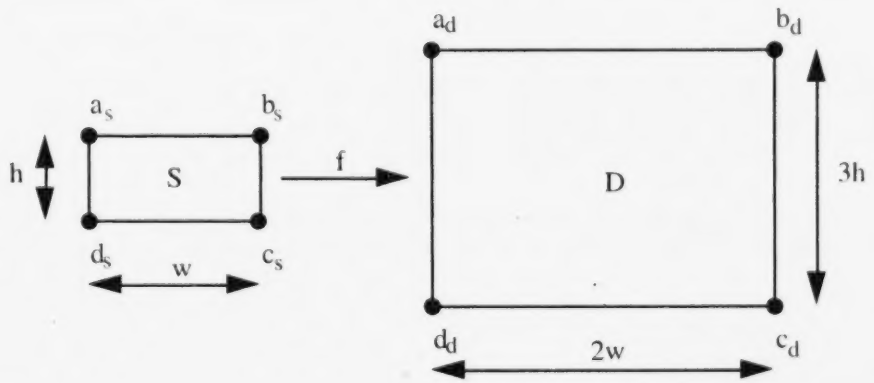


Figure 3.2: Mapping between rectangular images.

For an arbitrary shape that lies on a Cartesian Coordinate, generally there is no relationship between its edge and its interior points. It is difficult to find correspondences in both images. In fact, this is one of the most difficult problems in

any image warping algorithms. For example, if we would like to map S onto D in Figure 3.3, how do we determine the correspondences and scaling factors?

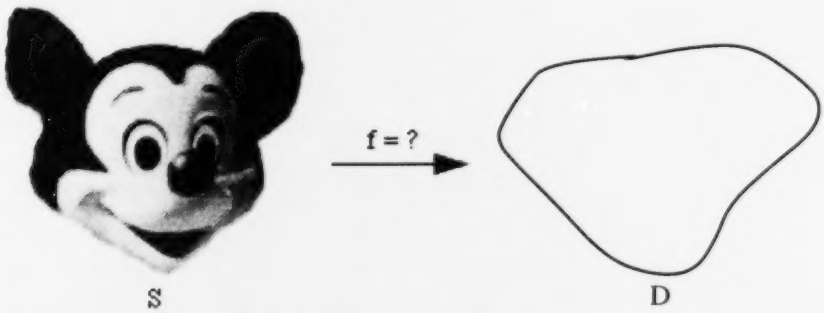


Figure 3.3: Mapping between arbitrary image shapes.

We could use the 2-pass method to map the rows first and then map the columns [Catmull and Smith80][Wolberg and Boult89]. As mentioned in the previous chapter, we could not use mapping functions that are classified as the rigid warping algorithms. However, the 2-pass method described in Chapter 2 can be treat as a free-form warping algorithm. The 2-pass method requires that both images must have the same dimension. Since the number of rows in the source and the destination may be different, the smaller image is scaled up so that both images have the same number of rows. Columns are treated similarly. When this scaling operation takes place, each image is treated as a rectangular image delimited by its smallest bounding box. In the first pass, we step along each row in the source image and map it onto the destination to produce an intermediate image. In the second pass, we go through each column in the intermediate image and map it onto the destination.

The problem with this approach is that the final image is not unique. The image depends on the order of the mapping. If we map columns first and then rows, we would have a different image than in the reverse case. This results in a direct consequence: an arbitrary image is not suitable for direct mapping in the Cartesian space.

Chapter 4

Arbitrary-Shapes Warping

There is only one existing warping algorithm for mapping between arbitrary shapes in the literature: the Skeleton-based method, which is described below.

4.1 Skeleton-based Warping

This section introduces the basic idea behind the Skeleton-based method. The next section gives the formal algorithm.

Wolberg extracts each image layer by a thinning process similar to peeling an onion [Wolberg88][Wolberg89] (see Figure 4.1). The thinning process creates a thinning path that reparameterizes the (x, y) coordinate system into a new (u, v) space, where u runs along the boundary of the image, and v runs radially inward across successive layers. When the source and destination images are thinned to their remaining skeletons, the (u_s, v_s) input coordinates are mapped to the output (u_d, v_d) by a 3-pass algorithm; subscripts s and d denote the source and destination space respectively. The first pass involves resampling each radial line in the source intermediate image to v_{max}, where v_{max} denotes the longest radial line. The result is a rectangular image which is fed to the second pass where the intermediate image is resampled along u so that u_s matches u_d. The third pass maps the radial line into its correct length along v. Finally, the destination image is traversed again using the same thinning paths, and the warped pixels are placed in their correct positions.

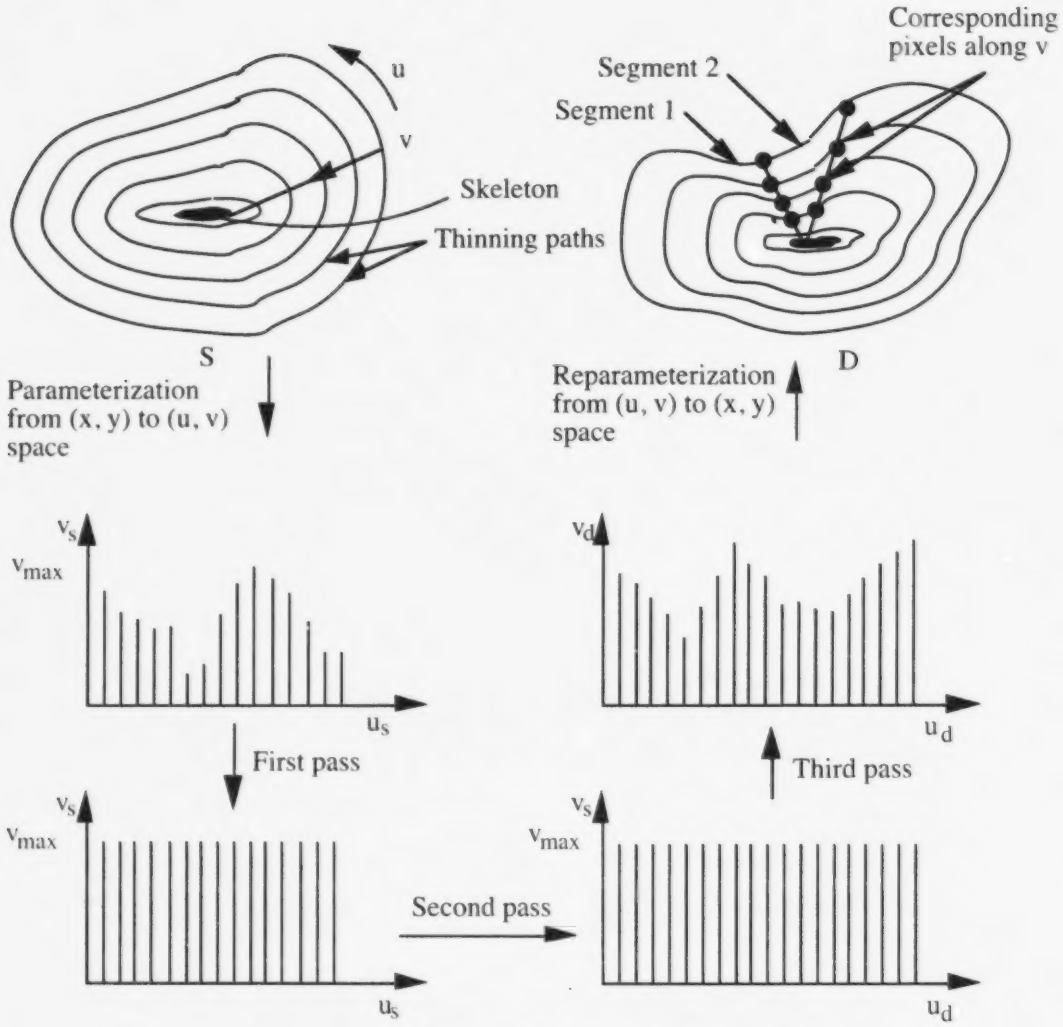


Figure 4.1: Conceptual Skeleton-based method.

4.1.1 3-pass Transformation

We mentioned that generally it is difficult to formulate a 2D transformation that maps images with arbitrary regions. Wolberg's 3-pass algorithm approximates the mapping function f in three 1D transformations. The idea is to reparameterize S and D from the Cartesian coordinate system to a new (u, v) system which provides a closer representation of arbitrary shapes. Image resampling is performed in (u, v) space. The

resulting image is reparameterized back to Cartesian coordinates. The three steps of the algorithm are outlined below. Step (1) corresponds to spatial transformation. Step (2) is the resampling process. Step (3) is the inverse of step (1).

(1) Convert S and D from Cartesian coordinate system, (x, y), to (u, v) space, using a transformation function g. This yields S' and D' respectively. The (u, v) representation of S and D simplifies the mapping. This step also involves normalizing S' in the direction of v for easy resampling in the second step.

(2) Transform S' to D' using a second transformation h.

(3) Transform D' to D using a transformation g^{-1} which is the inverse of g. Figure 4.2 shows the "decomposition" of function f into functions g, h and g^{-1}.

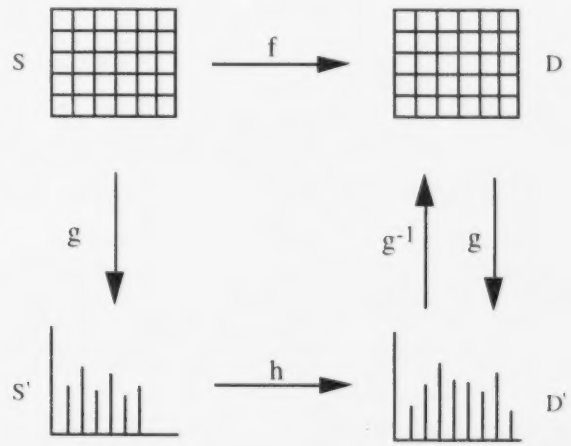


Figure 4.2: Skeleton-based transformation.

Figure 4.3 illustrates an example of the decomposition of a source image into a series of intermediate images. The elliptical image S is warped to a rectangular shape D.

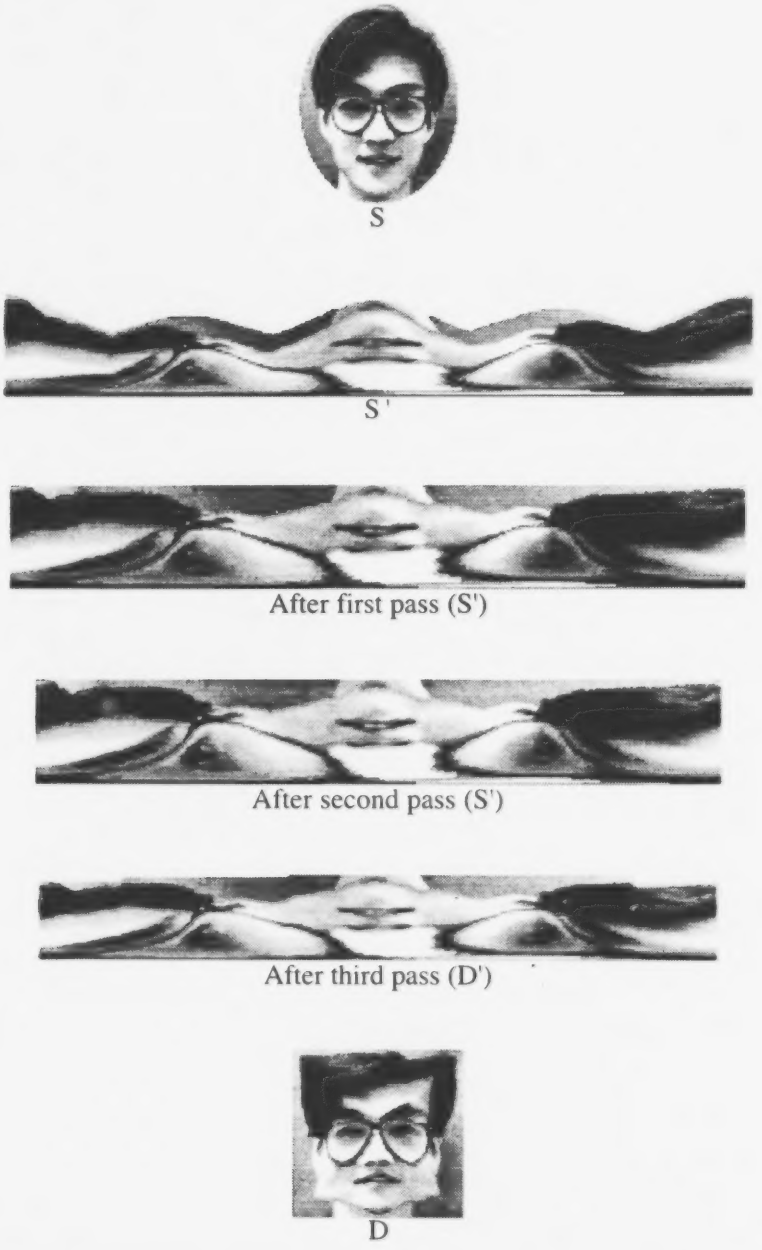


Figure 4.3: An example of intermediate images transformed by the Skeleton-based method.

4.1.2 Establishing Correspondences

The Skeleton-based method presents a few difficulties. The first is to define points that run radially. To establish a radial line, a point that lies on the first image layer needs to be aligned with a point in the second layer, and subsequent layers. Note that as the image is thinned one layer at a time, the entire radial line cannot be established immediately. It is established when all layers are thinned. Figure 4.1 shows that D has two radial lines with corresponding points. There is no definitive solution to choose those corresponding points. Wolberg chooses the top left-most point as the starting correspondence in each image. The correspondence changes to the first skeletal point as the image is thinned because this skeletal point is guaranteed to appear throughout the remaining of the thinning process. The second difficulty is in sampling each boundary layer. Since the outermost boundary layer of the image is clearly defined, u is defined. For this algorithm to work, each subsequent layer must have the same number of pixels as that of the outermost layer. That is, u must be the same across all layers. Generally, inner layers will contain fewer pixels than the outermost layer. The solution is to supersample the inner layers, which results in redundancy. When supersampling, the correspondences must be kept in the same positions, that is, on the same radial lines. This implies that when there are more than one correspondences in a layer, the sampling frequency for each segment is generally different and must be calculated based on the length of the corresponding outer segment (a segment of a layer is bounded by two correspondences). For example, Figure 4.1 shows that D has two segments in each layer. Consider segment 2 of the second outermost layer. This segment must be sampled to the same length as the corresponding segment in the outermost layer and this process must be repeated with segment 2 in each of the other inner layer, however, they all have different sampling frequencies. Moreover, each segment on the *same* layer may also have different sampling frequencies. Consider segment 1 and segment 2 of the second outermost layer. The sampling frequencies in these segments are different. This means we must keep track of all the segment lengths. One further restriction that this algorithm imposes is that both images must be topologically equivalent. That is, both images must contain the same number of holes or no holes at all. The correspondence problem becomes more complicated with the presence of holes.

4.2 **Radial Transform**

The motivations behind the Radial transform are to improve image quality over the Skeleton-based method, and to simplify the process of sampling and establishing correspondences. The algorithm is similar to that of the Skeleton-based method. The basic idea is to sample the images in the polar coordinates. Figure 4.4 illustrates the concept. First, we define an origin in the image and a radial line or path which starts from the origin and extends to the image boarder. This radial path is calculated using the Bresenham-line algorithm. We then sample the image radially according to this radial path. Next, we rotate the radial line by a certain degree and sample the image. This rotation produces a theta path. The theta path and radial path comprise the sampling grid. We keep rotating the radial line until it sweeps over the entire image for both the source and destination images. We then map each radial line from the source to the destination. Finally, we place each mapped radial line back into Cartesian coordinates using the same destination theta path. There are three stages of the Radial transform algorithm: spatial transformation, image resampling, and inverse spatial transformation.

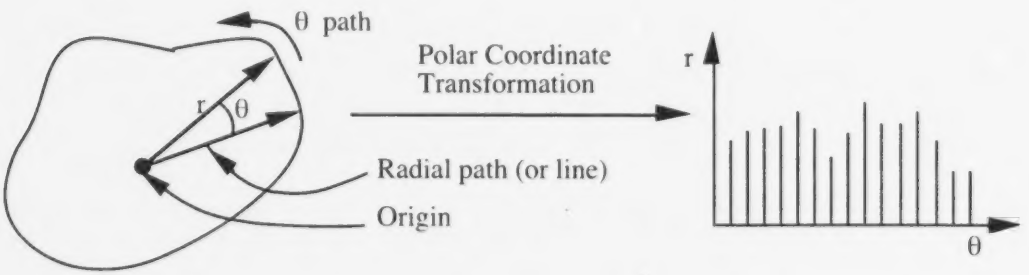


Figure 4.4: Conceptual Radial transform.

4.2.1 **Spatial Transformation**

The first stage is to reparameterize S and D from Cartesian coordinates into S' and D' of polar coordinates. The user needs to define only an origin and a radial sampling direction in both S and D. Before the radial sampling process takes place, S and D may

be upscaled when necessary. In reviewing the difficulties encountered by the Skeleton-based method, we find the following: First, in order to construct a radial line, a correspondence must be searched from one image layer to another. Second, the inner image layer is generally shorter than the outer layer, therefore, the inner layer must be supersampled so that the number of pixels sampled along u is the same in all layers. However, when supersampling a boundary segment, the correspondences must remain on the radial lines — therefore, the supersampling frequency must be calculated for each segment based on the segment length of the outer layer. This implies we must know the length of each segment. Third, both S and D must be topologically equivalent. Here, we tried to simplify the correspondence problem, the sampling process, and the topology problem that are present in the Skeleton-based method. The topology problem does not belong to the spatial transformation process and hence it needs a different section itself. It is discussed in the Mapping Between Radial Lines Section.

4.2.1.1 **Image Origin**

In Radial transform we need to calculate both r and θ to sample the image. Before r and θ can be calculated, we must define an origin. Unlike the Skeleton-based method of searching a correspondence in the image, we allow the user to define an origin as the correspondence. Since the origin is defined, both r and θ (v and u in Skeleton-based method) are defined as discussed later.

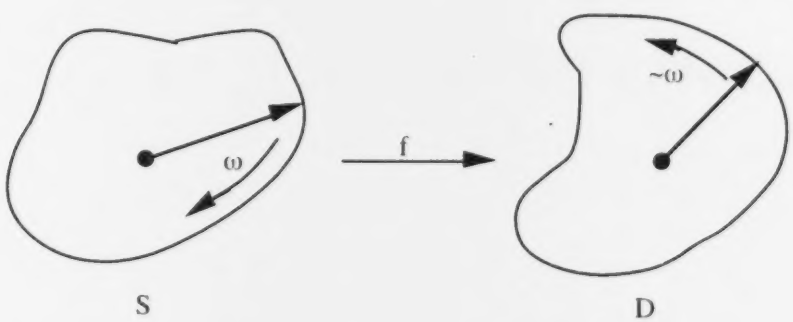


Figure 4.5: Image origins and sampling directions for Radial method.

The user defines an origin by drawing a radial line on each of the source and destination images. Figure 4.5 illustrates the idea. The user then defines the rotational or sampling direction as either clockwise (ω) or anticlockwise ($\sim\omega$). If the rotational directions in S and D are the same, the mapping may serve as a 2D rotation. If, however, the rotational directions are different, the mapping also serves as a lateral transformation.

4.2.1.2 **Correspondence Problem**

When we calculate a radial line from the origin to the image border, we establish all the correspondences along the radial line (or from one image layer to another layer in the Skeleton-based method). There is no need to search for correspondences — an advantage implicitly inherited in the Radial transform when calculating the radial lines. However, the following questions remain to be solved: "How many radial lines do we need in both the source and destination images to cover the image areas?" and "How do we establish radial-line correspondences between the two images?" Let us answer the second question first. To simplify the radial-line correspondence problem, we will assume that for each radial line in S' of polar coordinates, there must be a corresponding line in D' — then the mapping can go from S' to D' directly. This implies both θ_s and θ_d (or the number of radial lines) in S' and D' must be the same. Let us further assume that θ_s and θ_d are equal. This also answers part of the first question which can be rephrased as "How many degrees should we rotate the radial line for each rotation?" If the amount of rotation, θ, is too large, we may leave some area uncovered. If it is too small, we may redundantly process many radial lines. The optimal amount of rotation is implicitly defined by the number of pixels on the outermost image boundary. Note that the image lies on an integer lattice. Calculations should therefore be carried in integer coordinates as much as possible. Instead of using degrees, θ is defined by any two adjacent radial lines that cover the corresponding image area. This also means that θ may not be a constant value throughout the entire image. We previously assumed that θ_s equals θ_d, but the outermost boundary lengths of both images are usually different and therefore θ_s does not equal θ_d in practice. Before we solve this problem, let us first solve the "hole" problem which is described next.

4.2.1.3 Sampling Radial Lines

"Holes" may appear between adjacent radial lines. This is because our image lies on an integer lattice and a radial line must be approximated by a digital straight line. We used a Bresenham incremental straight line algorithm to approximate the radial line [Foley et al.90]. However, there is no guarantee that the two adjacent radial lines will cover every pixel between them. Instead of traversing along the outermost image boundary, we traverse the edges of the image bounding box one pixel at a time and draw a radial line from the origin. This approach eliminates "holes". Figure 4.6 illustrates the traversal.

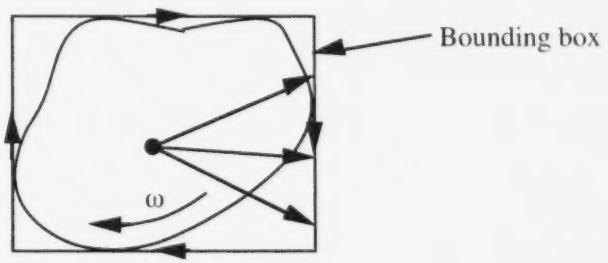


Figure 4.6: Bounding-box traversal.

4.2.1.4 Bresenham Line Algorithm

Consider the continuous line L and the digital line l in Figure 4.7. Detailed discussion on incremental algorithms can be found in [Foley et al.90]. The equation of line L that passes through the origin is

$$y = \frac{n}{m} x \tag{4.1}$$

where n and m are integers and $0 \leq n \leq m$. Lines that lie in other octants can be implemented by reflections on the x and y axes.

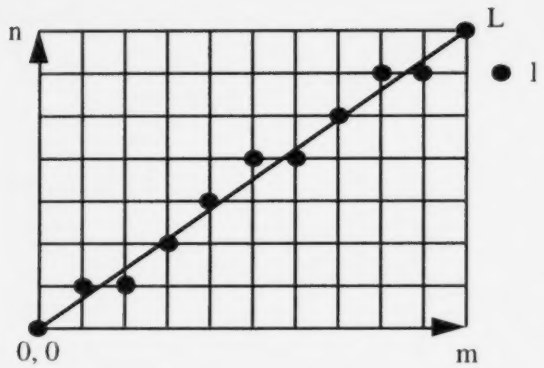


Figure 4.7: Real line L and digital line l on Cartesian coordinates.

Starting from a point P_0 on l, the next point, P_1, can be chosen so that it is closest to line L. We need to either choose the pixel immediately to the right of P_0 or the pixel to the right and upwards of P_0. For consistency, when tie occurs we always choose the pixel to the right of P_0. Then a point (x_i, y_i) with x_i, y_i integers belongs to the digital line l if

$$y_i = \frac{n}{m} x_i + e \qquad (4.2)$$

with error $|e| \leq \frac{1}{2}$ which is the vertical distance between the chosen pixel and the line L.

4.2.1.5 Elimination of "holes"

The bounding box has edges parallel to the axes of Cartesian coordinate. Let us consider the real lines L_1 and L_2 in Figure 4.8. L_2 is drawn from the origin to the edge of the vertical bounding box and one unit down from L_1. The equation of L_1 is

$$y_1 = \frac{n}{m} x \qquad (4.3)$$

and L_2 is

$$y_2 = \frac{n - 1}{m} x \qquad (4.4)$$

For $0 \le x \le m$, the vertical distance between L_1 and L_2 is

$$y_1 - y_2 = \frac{n}{m} x - \frac{n-1}{m} x \tag{4.5}$$

$$y_1 - y_2 = \frac{x}{m} \tag{4.6}$$

Since $\frac{x}{m} \le 1$ then

$$y_1 - y_2 \le 1 \tag{4.7}$$

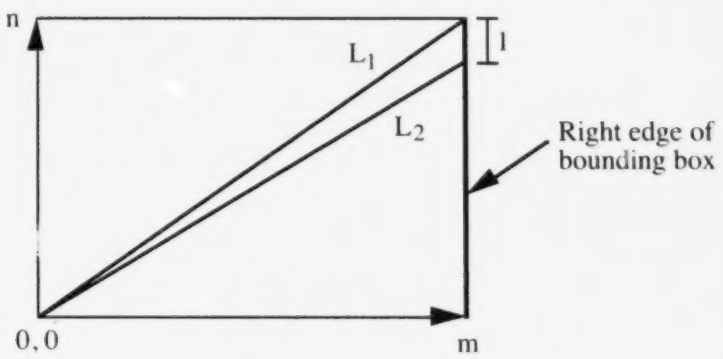


Figure 4.8: Real lines in real space.

Therefore, the vertical distance between L_1 and L_2 is always less than or equal to 1 for $0 \le x \le m$. The following gives a similar proof for digital lines. Consider Figure 4.9.

A point lies on l_1 if

$$y_1 = \frac{n}{m} x + e_1 \tag{4.8}$$

where $|e_1| \le \frac{1}{2}$ and x and y_1 are integers.

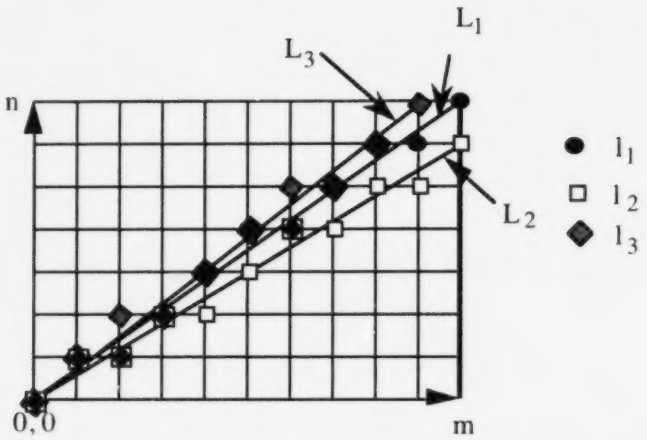


Figure 4.9: Three adjacent radial-digital lines that pass through the origin.

Consider an adjacent line l_2 which is drawn from the origin to the edge of the vertical bounding box and one pixel down from l_1. A point lies on l_2 if

$$y_2 = \frac{n-1}{m} x + e_2 \tag{4.9}$$

where $|e_2| \leq \frac{1}{2}$ and x and y_2 are integers.

Subtract equation (4.9) from (4.8)

$$y_1 - y_2 = \frac{x}{m} + e_1 - e_2 \tag{4.10}$$

Let us separate the proof in two cases: $x = m$ and $0 \leq x < m$.

Case 1: $x = m$, then

$$y_1 - y_2 = 1 \tag{4.11}$$

Case 2: $0 \leq x < m$

Since $\frac{x}{m} < 1$ and $(e_1 - e_2) \leq 1$, then

$$\frac{x}{m} + e_1 - e_2 < 2 \tag{4.12}$$

Since $y_1 - y_2$ is an integer, we have

$$y_1 - y_2 \leq 1 \tag{4.13}$$

Let us consider l_1 and l_3 where $n < m$ and $x \leq m - 1$.

A point lies on l_3 if

$$y_3 = \frac{n}{m - 1} x + e_3 \tag{4.14}$$

where $|e_3| \leq \frac{1}{2}$ and x and y_3 are integers.

Subtract equation (4.8) from (4.14)

$$y_3 - y_1 = (\frac{n}{m})(\frac{x}{m - 1}) + e_3 - e_1 \tag{4.15}$$

Since $(\frac{n}{m}) < 1$ and $(\frac{x}{m - 1}) \leq 1$, then

$$(\frac{n}{m})(\frac{x}{m - 1}) < 1 \tag{4.16}$$

and we have

$$y_3 - y_1 \leq 1 \tag{4.17}$$

Hence, there are no "holes" between the adjacent radial lines.

The "hole" problem is solved by redundantly sampling the image. This supersampling process is much simpler than that used in the Skeleton-based method. However, when traversing the edges of the bounding box, we want only the foreground pixels covered

by the radial lines to be sampled. To avoid calculating intersections between the image boundaries and the radial lines, we index a boundary map to test if the radial lines cover foreground pixels. A boundary map, which is the same size as the image, specifies which pixels are within image boundaries. The boundary map is covered in the Mapping Between Radial Lines Section.

When traversing S and D, the Cartesian coordinate of each pixel on a radial path is stored in a look-up table. After S' is mapped to D', pixels from D' will be placed back in D. For each pixel in D', we simply index into the look-up table to find its Cartesian coordinate and place it into its correct position in D. The look-up table avoids recalculating the positions of pixels that lie on radial paths in D.

4.2.1.6 **Radial-Line Correspondences**

We now come back to the question of how to make θ_s equal θ_d. We have chosen an easy method to radially sample the image and eliminate "holes". This method also simplifies the process of making θ_s equal to θ_d. In order for θ_s to be equal to θ_d, we mentioned that the outermost image boundary lengths for both images must be the same. Since we have chosen the bounding box as the traversal criteria, the perimeters of the bounding boxes in both images must be equal. If they are different the smaller images are scaled to match the largest dimensions. For example, if S is 3*4 and D is 5*2, then both images are scaled to 5*4 before sampling. Note that we need to scale only the boundary map of D not the image D. Bilinear interpolation is used if scaling image S is necessary. The disadvantage is the increase in computational time. However, the time is bounded by the largest dimension among the two images.

4.2.2 **Image Resampling**

Stage two involves resampling each radial line from S' to D'. Fant's 1D resampling algorithm is used because it is both fast and avoids alias artifacts [Fant86]. Since both S and D have the same dimensions, they have the same number of radial lines. After radial sampling, each radial line in S' has a corresponding line in D'. The resampling process simplifies to mapping each radial line in S' to the corresponding line in D'. Before we describe Fant's algorithm, let us discuss bilinear interpolation used for

scaling images in the first stage, and linear interpolation needed for the radial sampling process.

4.2.2.1 **Bilinear Interpolation**

Bilinear interpolation uses the four "closest neighboring" pixel values to interpolate the new value. Given four points that lie on a rectangular grid, (x_0, y_0), (x_1, y_1), (x_2, y_2), and (x_3, y_3), and their intensity values v_0, v_1, v_2, and v_3 respectively, any intermediate coordinate value $V(x_0 + x', y_0 + y')$, where $0 < x', y' < 1$ can be computed as

$$V(x_0 + x', y_0 + y') = a_0 + a_1x' + a_2y' + a_3x'y' \tag{4.18}$$

where a_i are coefficients that can be computed from the following matrix expression

$$\begin{vmatrix} 1 & x_0 & y_0 & x_0y_0 \\ 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{vmatrix} = \begin{vmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{vmatrix} \tag{4.19}$$

Solving for a_i and substituting them into (4.18) yields

$$V(x_0 + x', y_0 + y') = v_0 + (v_1 - v_0)x' + (v_2 - v_0)y' + (v_3 - v_2 - v_1 + v_0)x'y' \tag{4.20}$$

To achieve computational efficiency, bilinear interpolation can be represented by 2-pass transformation. In the horizontal pass, each row in the image can be computed to produce an intermediate image.

$$v_{01} = v_0 + (v_1 - v_0)x' \tag{4.21}$$

$$v_{23} = v_2 + (v_3 - v_2)x' \tag{4.22}$$

In the vertical pass, each column in the intermediate image can be interpolated to produce the final image.

$$V(x_0 + x', y_0 + y') = v_{01} + (v_{23} - v_{01})y' \tag{4.23}$$

4.2.2.2 Linear Interpolation

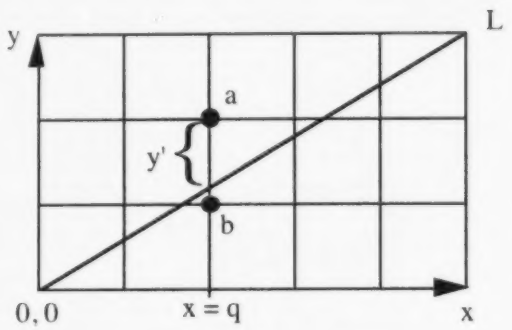


Figure 4.10: Linear interpolation for a digital line.

When using the Bresenham-line as the sampling grid, pixels are point samples of S in polar coordinates. Errors are introduced when we approximate the real radial line with a Bresenham-line and point sample the "closest" pixels. This error can be minimized by interpolating the two "closest" pixels that lie on the opposite side of this line. Consider line L with x = q in Figure 4.10. The Bresenham-line algorithm that approximates L may choose either pixel a or b as the closest one. However, L actually passes in between pixels a and b. A better result can be obtained by interpolating pixels a and b. Given that the intensity values of pixels a and b are v_a and v_b respectively, a linear interpolation would yield $y'v_a + (1 - y')v_b$, where $0 \leq y' \leq 1$. To achieve better speed performance we do not use interpolation when radially sampling the image.

4.2.2.3 Fant's Resampling Algorithm

Fant presents a fast 1D resampling algorithm that combines image reconstruction and antialiasing to map all input pixels onto the output, both in scanline order [Fant86]. The input is consumed at a rate determined by the spatial mapping function and any of the following three situations may arise:

(1) Only a fraction of the input pixel is consumed while completing the output pixel. The remaining portion of the input pixel will be used to fill in the next output pixel.
(2) The entire input pixel is consumed without completely filling the output pixel.
(3) The entire input pixel is consumed while completing the output pixel.

Each of these situations is described shortly.

The block diagram of Fant's resampling algorithm is illustrated in Figure 4.11.

• SIZFAC is the scaling factor

• INSFAC(= $\frac{1}{\text{SIZFAC}}$)is the inverse of the scaling factor; it determines how many input pixels contribute to each output pixel,

• INSEG is the portion of the current input pixel which remains to contribute to the next output pixel.

• OUTSEG is the portion of the current output pixel which remains to be filled.

• Pixel is the intensity value of the current input pixel.

• Factor can have a value of either INSEG or OUTSEG.

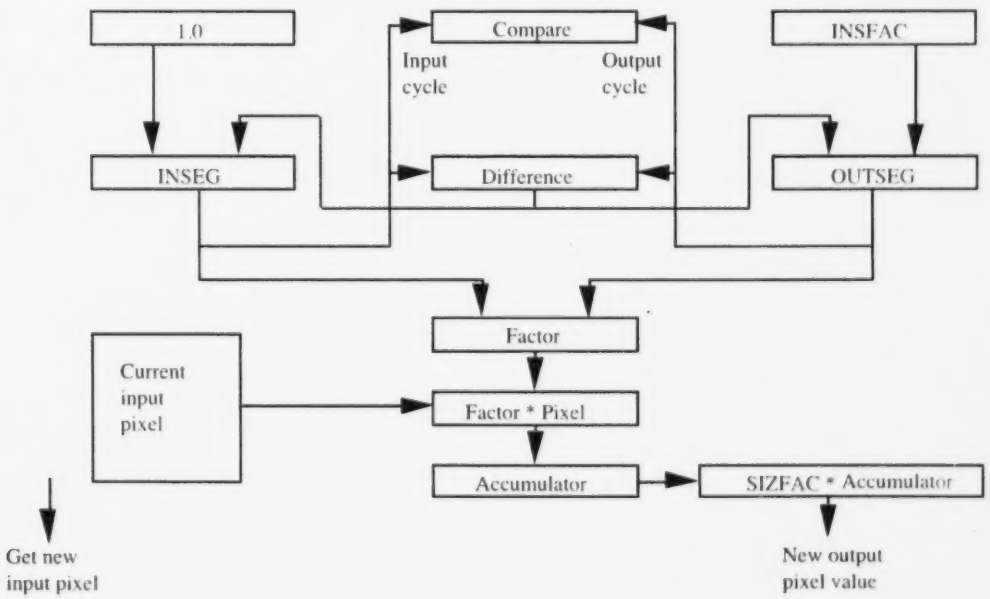


Figure 4.11: Fant's 1D resampling algorithm (redrawn) [Fant86].

The process begins by comparing the values of INSEG and OUTSEG. If INSEG is larger than OUTSEG, an output pixel will be fully filled. This corresponds to the first situation described earlier. Factor gets the value of OUTSEG. Pixel is multiplied by Factor and the result is added to the Accumulator. Since the OUTSEG portion of the input pixel has been used, INSEG is decreased by the value of OUTSEG. Since the entire output is filled, OUTSEG is reinitialized to INSFAC and Accumulator is multiplied by INSFAC, yielding the value of the current output pixel. Accumulator is initialized to 0 and the process repeats for comparison between INSEG and OUTSEG.

If OUTSEG is larger than INSEG, the entire current input pixel will be consumed. This corresponds to the second situation. Factor gets the value of INSEG. Pixel is then multiplied by Factor and the result is added to the Accumulator. Since the INSEG portion of the output pixel is filled, OUTSEG is decreased by the value of INSEG. Since the entire input pixel is consumed, INSEG is reinitialized to 1.0, and the next input pixel is fetched. Then the process repeats.

If INSEG equals OUTSEG, the entire input pixel will be consumed and the whole output pixel will be filled. This corresponds to the third situation. In this case, only one event can occur at a time, but both events will occur. Either event can be chosen to occur first, and the algorithm will proceed correctly.

4.2.3 **Inverse Spatial Transformation**

Stage three converts D' from polar coordinate representation to Cartesian coordinate representation D and when necessary, downscales it to the same dimension as the original destination image using bilinear interpolation. Earlier we noted that when sampling D in stage one, we stored the Cartesian coordinate of each pixel on a radial path in a look-up table. When placing pixels from D' back into D, we index into the look-up table to find their Cartesian coordinates.

The complete Radial algorithm is outlined below. Figure 4.12 shows the block diagram of the Radial transformation.

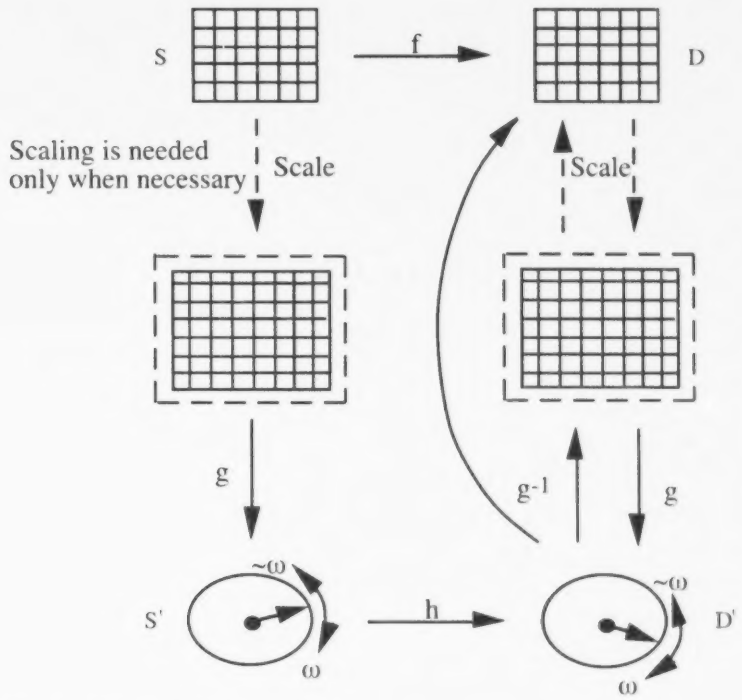


Figure 4.12: Radial transformation.

Stage 1

(1) Upscale the source image S and its corresponding boundary map. This is necessary only if the dimensions of the bounding box of the source image are smaller than that of the destination image. Bilinear interpolation is used if scaling is necessary.

(2) Upscale the boundary map of destination image D. This is necessary only if the dimensions of the bounding box of the destination image are smaller than that of the source image.

(3) Convert S and D from Cartesian coordinate system, (x, y), to polar coordinates, (r, θ), using a transformation function g. This yields S' and D' respectively. The polar coordinate representation of S and D simplifies the process of establishing correspondences and hence simplifies the resampling stage. This step involves calculating the sampling grids for S and D and samples S radially.

Stage 2

(4) Transform S' to D' using a second transformation h. This step involves resampling each radial line of S' to the length of the corresponding line in D'.

Stage 3

(5) Convert D' to D from polar coordinate system to Cartesian coordinates using a transformation function g^{-1} which is the inverse of g.

(6) Downscale the final image. This is necessary only if the size of the final image is larger than that of the original destination image. Bilinear interpolation is used if scaling is necessary.

Figure 4.13 shows the decomposition of a source image into a series of intermediate images. The elliptical image S, with its origin superimposed, is transformed into a rectangular image D. Both images have the same rotational directions.

S

S'

D'

D

Figure 4.13: An example of the intermediate images transformed by the Radial transform.

Particular attention is needed when performing step (6). We chose bilinear interpolation for downscaling the final image. If the foreground pixels of the final image do not cover the entire area of the bounding box, bilinear interpolation will introduce errors along the boundary of foreground pixels. This is illustrated in Figure 4.14. This error occurs at the vicinity of image boundary because the bilinear interpolation algorithm may choose a background pixel as one of its four "closest neighbor" pixels in the interpolation process. However, this error can be easily corrected. When sampling the pixels in step (6), we may use the boundary map to sample and interpolate only the foreground pixels. In the worst case, bilinear interpolation of pixels along the image boundary degrades to point sampling.

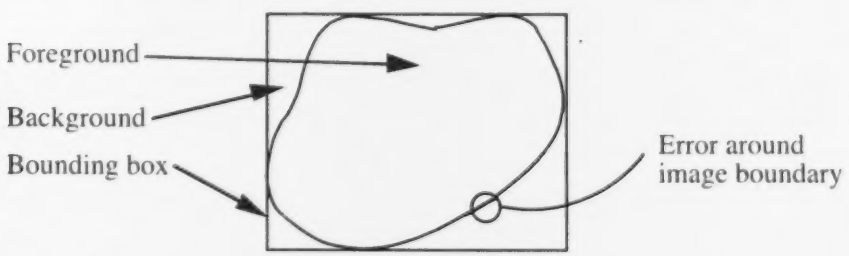


Figure 4.14: Interpolation error around image boundary.

4.3 Mapping Between Radial Lines

Although the Radial transform provides a convenient way to map images that contain holes, the warped image may have discontinuity at the breaking of radial lines, as we shall see shortly. The user can define any closed free-hand drawn shapes for mapping. This includes shapes having any number of holes, islands that are inside the defined shapes, and object boundaries that overlap themselves (Figure 3.1). Recall that we would like to map any image onto any other image even if their topologies are different. To perform the mapping correctly, we must map all the input pixels to the output pixels. For example, Figure 4.15 shows the source image with a hole, and the destination is any hand drawn shape without holes. The lines in the images represent the current radial lines to be mapped. Note that the source image has two line segments of pixels mapped to a single line segment in the destination.

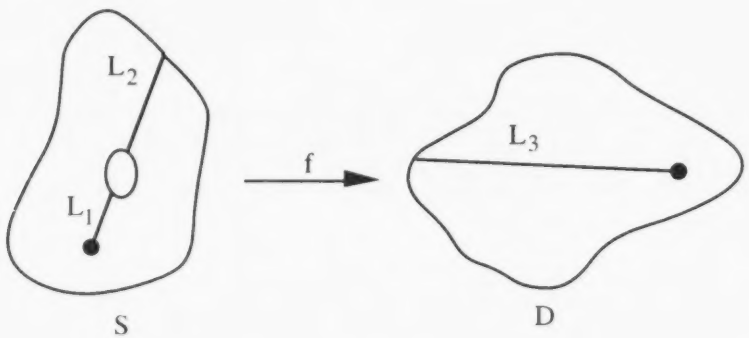


Figure 4.15: The first example for mapping of a radial line between two arbitrary topological different image shapes.

To map all the input and output line segments completely, we must know the length of each segment for both the source and destination images. This is found by calculating the intersections between the radial line and the image boundaries. For the example above, assume the calculated lengths of the line segments are L_1 and L_2 for the source image and L_3 for the destination. If we normalize the line segments, we can easily compute the portion of the source pixels that map to the destination. In this case L_1 will

be mapped to $\frac{L_1}{L_1+L_2}$ portion of L_3. And L_2 will be mapped to $\frac{L_2}{L_1+L_2}$ portion of L_3. Both the source and destination line segments are completely mapped.

However, if there are many line segments in both images, the calculations become quite tedious because one line segment may be mapped to two or more line segments (Figure 4.17). Similarly, two or more line segments may be mapped to a single line segment. We must know exactly the position of the last mapped segment in both S and D. Fortunately, we can simplify all the calculations above by noting that each line segment in the image can be extracted to form a *single* line segment. Figure 4.16 illustrates the extracted line segments for Figure 4.15. For simplicity, only the lengths of the segments are shown, the actual pixel values are not included. Figure 4.16 shows how the input image pixels of length L_1 and L_2 are extracted to form a single line segment. This single line segment is then mapped to the output length L_3. The dotted curves show the mapping positions from the source to the destination.

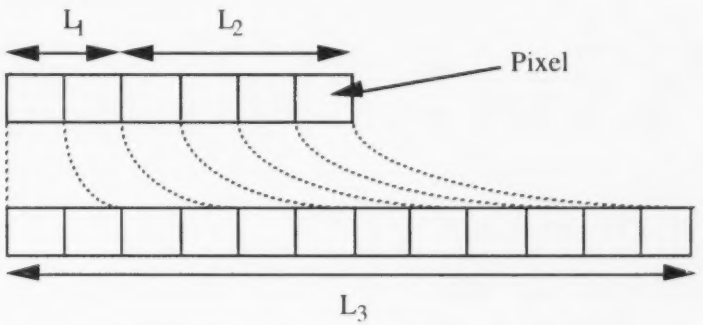


Figure 4.16: Mapping input image pixels from two line segments to one for Figure 4.15.

After obtaining all the image boundary points from the user, we construct a boundary map for each image. The boundary maps specify the boundaries of the user defined shapes and have the same dimensions as those of S and D. Instead of calculating the intersections between the radial line and the image boundaries to sample the foreground pixels, we step along the radial line and index the boundary map. This will tell us if the pixel belongs to the foreground or the background and we can decide if sampling is necessary. We then sum the number of pixels sampled and compute the mapping

functions. The mapping function becomes a simple ratio between the input length and output length (which is $\frac{L_1+L_2}{L_3}$ for our example). Figure 4.17 shows another example where there are two line segments in S and three in D. The mapping of the input to output pixels is shown in Figure 4.18.

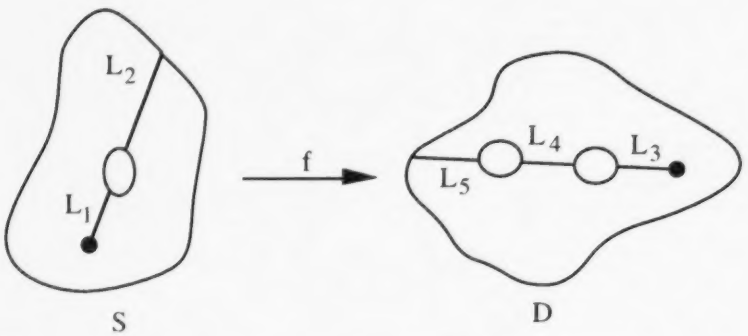


Figure 4.17: The second example for mapping of a radial line between two arbitrary topological different image shapes.

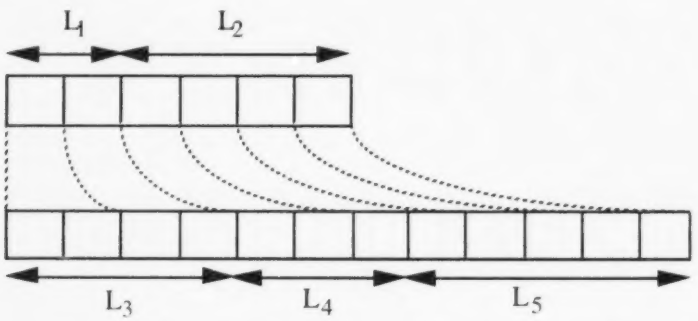


Figure 4.18: Mapping input image pixels from two line-segments for Figure 4.17.

Although the Radial transform allows the mapping of topologically different images, the breaking of radial lines at holes may not necessarily preserve the smoothness of the mapping in some cases. Examples are provided in the Gallery Section. The Radial transform simplifies the correspondence problem and the resampling process, however, the source image may not be "evenly" stretched to take the shape of the destination. Consider Figure 4.19 where image S is mapped into a rectangular image D with both images superimposed. The Skeleton-based method stretches each radial line in S

evenly from the skeleton towards the boundary of D. This process is reminiscent of the action of a small volume of oil as it tries to spread all the available area of the container. On the other hand, depending on the image shapes and the positions of the origins, the Radial transform may not stretch S "evenly". Figure 4.19 shows a possible origin in S where the Radial transform would not produce "even" stretching.

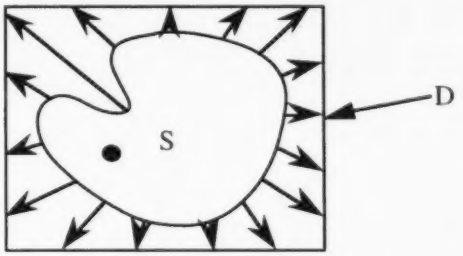


Figure 4.19: "Even" image stretching.

4.4 Theoretical Analysis

There are two analyses that are important for warping algorithms. The first concerns the quality of the warped image; the second, the speed of the algorithms. The theoretical analysis of the Skeleton-based method and Radial transform are given below. We also include the memory requirement for the Radial transform although no analysis is given for the Skeleton-based algorithm.

4.4.1 Image Quality

The Skeleton-based algorithm introduces errors in two places: the resampling process and the order of this resampling process. After the first step of the Skeleton-based method, the intermediate image is resampled along v in the first pass. For the purpose of discussion, we use a simplified version of the Skeleton-based method. We do not resample the radial lines along v in step (1), but only resample the intermediate image along u. This is valid for the example in Figure 4.20a.

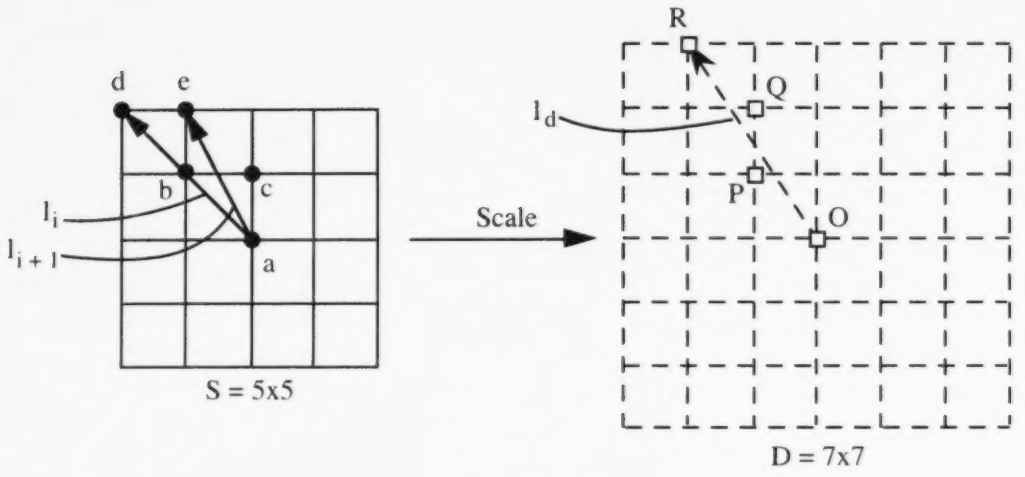


Figure 4.20a: Enlarge image S to D with radial lines and sampled pixels.

Consider Figure 4.20a. We enlarge S = 5*5 to D = 7*7. The source image is placed on a solid grid whereas the destination image is placed on a dashed grid. Since the images are perfect squares and the Skeleton-based algorithm thins the images from the outermost boundary and works towards the center (which is the skeleton), all radial lines in S have the same length so there is no need to resample along v. For the Radial transform we simply choose the centers of S and D as the origins. Both methods will then sample S and D exactly the same way. Let us consider two adjacent radial lines in S and one in D, and their sampled pixels. Figure 4.20b shows the two images superimposed (they are not illustrated in the same scale). The sampling grid of D is embedded in S for "even" interpolation. Also note that l_d is in between the adjacent lines l_i and l_{i+1}.

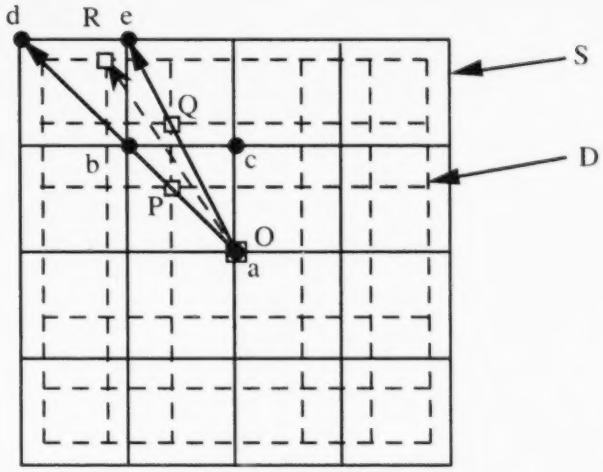


Figure 4.20b: Superimposed images S and D with radial lines and sampled pixels.

The pixels sampled for l_i and l_{i+1} are shown in Figure 4.21a. Due to the discrete image domain, the middle pixel of l_{i+1} indicates that either pixel b or c is chosen. When the second pass of the Skeleton-based method resamples u_s to u_d, it does so along the image layers (in this example, u_s is 16 and u_d is 24). For linear interpolation, assume two columns from S' (Figure 4.21a) will be used for interpolating each column in D' (Figure 4.21b). Readers may have already notice this in Figure 4.20b where l_d is in between l_i and l_{i+1}. Consider the bottom row of Figure 4.21a and 4.21b. Interpolation of l_i and l_{i+1} to produce l_d uses pixel a only. For the middle row, the pixels used for interpolation are b and b, or b and c. We are only interested in the pixels chosen for interpolation, therefore, only one b is included instead of b and b. For the top row, pixels d and e are used. The result is shown in Figure 4.21b. The line l_d is then resampled from v_s (= 3) to v_d (= 4). The result is shown in Figure 4.21c.

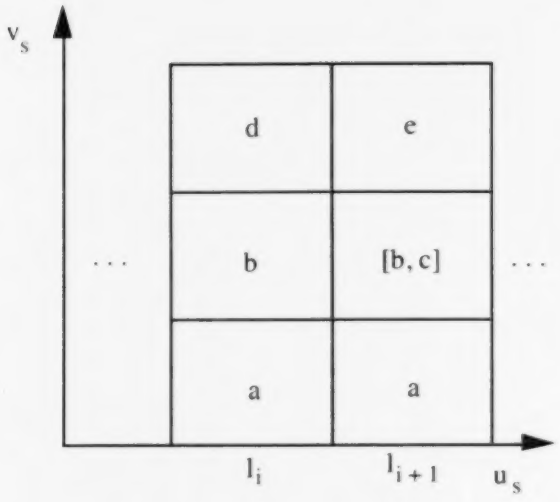


Figure 4.21a: Sampled pixels for Skeleton-based method.

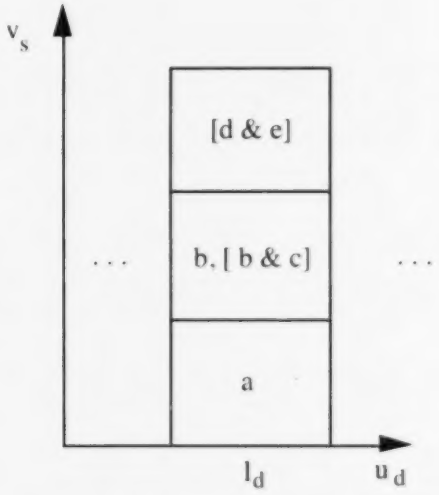


Figure 4.21b: Pixels after resampling along the u-axis.

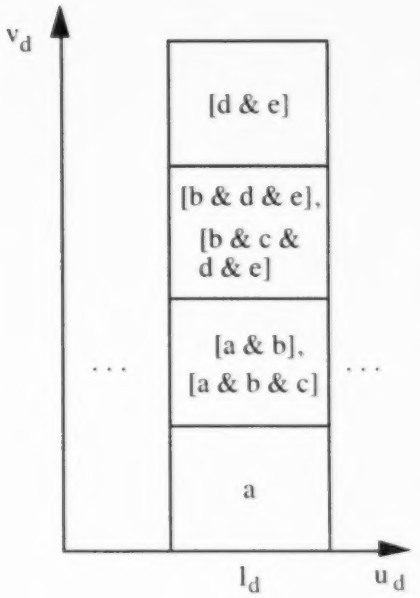


Figure 4.21c: Pixels after resampling along v-axis.

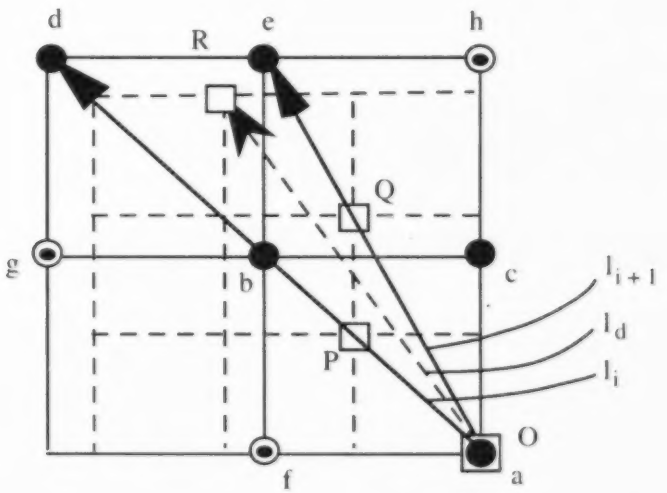


Figure 4.22: Errors in sampling for Skeleton-based method.

When we convert the image from the (u, v) to (x, y) space in the Skeleton-based method we obtain the final image. The errors of the interpolation are shown in Figure 4.22. Consider pixels O, P, Q, and R in the final image. For interpolation, the Skeleton-based method chooses pixels [a], {[a, b] or [a, b, c]}, {[b, d, e] or [b, c, d, e]}, and [d, e] respectively (Figure 4.21c, 4.22) whereas the Radial transform chooses pixels [a], [a, f, b, c], [c, b, e, h], and [b, g, d, e] respectively (using bilinear interpolation). These input and output pixels are summarized in Table 4.1.

Table 4.1: Input pixels sampled for the Skeleton-based and the Radial methods.

	Skeleton-based	Radial
O	[a]	[a]
P	[a, b] or [a, b, c]	[a, f, b, c]
Q	[b, d, e] or [b, c, d, e]	[c, b, e, h]
R	[d, e]	[b, g, d, e]

With Cartesian coordinates, the Skeleton-based method *does not* chose the closest neighbors of O, P, Q, and R for interpolation. Moreover, the contribution of the source pixels is not in the correct proportion. For example, when u_s is resampled to u_d, the scaling factor is 3/2 (= 24/16). However, this distribution of pixel weights is not justified in the Cartesian coordinates where the scaling factor is 7/5. Therefore when viewing the final image in the Cartesian coordinates, the result will not be as good as that produced by the Radial transform.

Note that both methods also interpolate an image radially which is "inappropriate" when the image is viewed in the Cartesian coordinates. The Skeleton-based method performs this interpolation in the first and third passes which is worst than that of the Radial transform where this interpolation is only performed once in stage 2 step (4). This "inappropriate" interpolation is inherited in both methods.

By examining the Skeleton-based method more closer, we note that l_d first chooses the closest lines l_i and l_{i+1} in S. It then chooses 3 or 4 pixels which lie on l_i and l_{i+1} and are closest to a point on l_d. This can be generalized as follows: the Skeleton-based algorithm chooses radial lines l_s from the source image that are closest to the radial lines

l_d in the destination image. The pixels that lie on l_d are the interpolation of the closest pixels that lie on l_s.

In the first step of the Radial transform, the original image may be scaled up if necessary. This scaling operation is performed in the Cartesian space. The image is then radially sampled (in step (3)). Next, each radial line in the intermediate image is resampled to the destination length. The last step involves scaling down the final image if necessary, again in Cartesian space. In both the first and last steps, bilinear interpolation is used. The four "closest neighbor" pixels are chosen in Cartesian space. Therefore, the Radial method performs image reconstruction in Cartesian space as many as possible. On the other hand, the Skeleton-based method performs all image reconstruction in (u, v) space where interpolations are not "appropriate" when images are viewed in Cartesian coordinate.

There is a situation where Radial transform will not generate a higher image quality than that produced by the Skeleton-based method. If the source and destination images are very thin, if the Radial transform introduces the breaking of radial lines in the image D', and if the image D' needs to be downscaled to D after the mapping, then the Radial transform will not produce a high quality image. For example, imagine mapping an image of a cigarette to an image of a thin Chinese dragon. Assume the origin in each image is at the centriod. Consider the image D' at the vicinity of the breaking of radial lines introduced by the mapping. When downscaling D' using bilinear interpolation, the breaking of radial lines in D' will introduce errors to the neighboring pixels.

The second error concerns with the order of the resampling process. There is an important difference between the two methods. In general, interpolation *before* sampling produces higher image quality than sampling and then interpolating the samples [Wolberg90]. The Radial transform performs image reconstruction before radial sampling and the Skeleton-based method performs interpolation after sampling. For the Radial transform, the final image quality partially depends on the interpolation algorithm used for upscaling and downscaling the images.

4.4.2 **Speed Performance**

Speed is an important criterion in evaluating an image warping algorithm. We will show the upper bound speed limit for the Radial transform. Speed is directly proportional to the total cost of performing each operation in the three stages of the Radial algorithm. The cost of each operation is directly proportional to the image area. For example, scaling S in step(1) is proportional to the largest image area in S and D. Let this area be $A_1 = wh$, where $w = \text{maximum}(w_s, w_d)$ and $h = \text{maximum}(h_s, h_d)$. In step(3) we radially sample S. The largest area covered is $A_2 = 2wh - w - h - \text{minimum}(w, h) + w(w-1)/2 + h(h-1)/2 - 1$. This area is derived by having the origin at any corner of the four corner points of the bounding box of S. The costs of all the operations are listed in Table 4.2.

Table 4.2: Cost of performing all steps of Radial warping.

cost of scaling S	$Sc(A_1)$
cost of scaling S's and D's boundary map	$2Sc(A_1)$
cost of calculating polar coordinates	$2Pc(A_2)$
cost of sampling S to S'	$Sp(A_2)$
cost of mapping S' to D'	$Mp(A_2)$
cost of placing D' onto D	$Sp(A_2)$
(This cost is the same as that of sampling S to S')	
cost of scaling D	$Sc(A_1)$
Total cost:	$= 4Sc(A_1) + 2Pc(A_2) + 2Sp(A_2) + Mp(A_2)$

Note that the scaling for boundary maps using a simple scaling algorithm can be much faster than scaling the image using an interpolation algorithm. We are only interested in whether the boundary map specifies a foreground or background pixel. We are not interested in the accuracy of this value, hence interpolation is not necessary for scaling boundary maps. For most images, the Skeleton-based method may run faster than the Radial transform. In the Radial transform, both S and D may need to be scaled up, and the user can place the origin anywhere in the image. The position of this origin directly influences the area sampled; the intermediate image can be very large (A_2). In the Skeleton-based method the intermediate image area is approximately determined by the

product of the length of the outermost layer and the number of layers when the skeleton is reached.

4.4.3 Memory Requirement

Assuming the images and their boundary maps have already been scaled up, the memory requirement for Radial transform is given in Table 4.3. The area A_1 and A_2 are given in the Speed Performance Section.

Table 4.3: Memory costs

storing of S and D	$2Im(A_1)$
storing of S's and D's boundary maps	$2Im(A_1)$
storing of look-up tables	$2Lut(A_2)$
storing of S' and D'	$2Im(A_2)$
Total memory costs:	$= 4Im(A_1) + 2Lut(A_2) + 2Im(A_2)$

4.5 Radial-Axes Warping

The Radial-axes transform is a combination of both the Radial spatial transformation and the Skeleton-based 3-pass technique. This method gives the user more local control of the warping process over the Radial transform technique. Since we use the 3-pass algorithm for image resampling, the image quality will be similar to that used in the Skeleton-based method. The Radial-axes method uses two axes to divide an image into four regions for both images S and D. A region in S is then mapped to the corresponding region in D using the 3-pass algorithm.

4.5.1 Spatial Transformation

The spatial transformation of the Radial-axes method is similar to the Radial method except that two axes are used to define the origin and the rotational direction of an image.

4.5.1.1 Image Origin

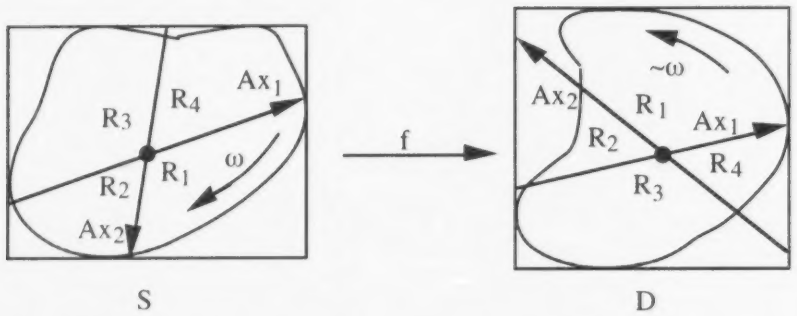


Figure 4.23: Image division for Radial-Axes method.

The user defines an origin by drawing two non-parallel axes on each image. The origin is the point where the two axes cross. Figure 4.23 shows the bounding box, the rotational direction, image boundary, the axes, and the divided regions for each image. The axis is defined by a line passing two user selected points. Each axis is a directed line with its direction pointing from the second user point towards the first. The two axes are labeled Ax_1 and Ax_2 depending on the order in which they are defined. The axes divide an image into four regions R_1, R_2, R_3, and R_4. The rotational direction is defined differently than that of the Radial method. Instead of explicitly providing the rotational direction by the user, the two axes together define a rotational direction, either clockwise (ω) or anti-clockwise ($-\omega$). The rotational direction is defined by the smallest angle made by rotating Ax_1 onto Ax_2 (with the origin as center of rotation) so that both axes coincide with each other and point in the same direction. The rotational direction serves two purposes: the regions are numbered sequentially according to the direction of rotation, and the radial sampling direction is the same as the rotational direction. In Figure 4.23 numbered regions in S are mapped to the corresponding regions in D. If the rotational directions in S and D are the same, the mapping may serve as a 2D rotation. If, however, the rotational directions are different, the mapping also serves as a lateral transformation.

The first step is to reparameterize S and D from Cartesian coordinates into S' and D' of polar coordinates. This stage involves sampling S and normalizing it in the direction of

r-axis. The process begins in region R_1 of S by traversing the bounding box. Starting from the first axis, we calculate a radial path from the origin to the edge of the bounding box. Only the foreground pixels are sampled. We then increment theta clockwise or anti-clockwise to calculate the next radial path. When the entire area of region R_1 is sampled, the process continues for R_2, R_3, and R_4 until the whole image is covered.

4.5.2 **3-pass Transformation**

When the image is divided into regions there is no guarantee that each image region is rectangular. If we want to use the Radial transformation, we must form a bounding box around each region and treat it as an image. Instead, we used the Skeleton-based 3-pass transformation.

First pass

After sampling a region in S, each column on the θ-axis is resampled along the r-axis to the longest radial path in this region. All columns are supersampled at a sampling rate that exceeds the Nyquist rate. The result is a rectangular image suitable for resampling in the second pass.

Second pass

The second pass resamples the resulting rectangular image along the θ-axis so that the number of radial lines in S' matches that in D'. Each row in S' can be resampled independently using a lD transformation.

Third pass

The intermediate image produced in the second pass has the correct number of radial lines, but not the correct length along the r-axis. Each column in S' is resampled to the dimension of the corresponding column in D'.

4.5.3 **Inverse Spatial Transformation**

As in the Radial transform, when traversing D for sampling, the position of each pixel on a radial path is stored in a look-up table. When placing each column of D' back in D, we traverse the corresponding radial path and index the look-up table to obtain the positions in Cartesian coordinates.

Chapter 5

Performance Comparisons

In this section, the comparisons of image quality are given first, followed by the comparisons for speed performance. Our experiments showed that the Radial transform produces image quality superior to that produced by the Skeleton-based method. For the speed performance in general, the Skeleton-based method transforms an image faster than that of the Radial transform, with a few exceptions.

5.1 Signal-to-noise Ratio Comparisons

For an m*n image, the image quality is measured by the mean-square signal-to-noise ratio (SNR) which is defined as follows:

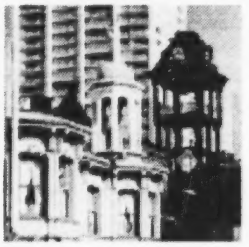
$$SNR = \frac{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} v'^2(x,y)}{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} (v'(x,y)-v(x,y))^2} \qquad (5.1)$$

where v(x, y) is the pixel value of the original image and v'(x, y) is the warped pixel value.

We categorize images into two groups. The first group contains images with high frequencies in the frequency domain. Typical images in this group exhibit many sharp edges in the spatial domain. The second group contains images with low frequencies in

the frequency domain. These images do not have many sharp edges; the images appear to be smooth over an area. Our tests include three images in each group. Within each group, images are arranged from high frequencies to low frequencies, relative to that group. The test images in the first group are included in Figure 5.1a and those in the second group are shown in Figure 5.1b. These images are shown at half of their original size which is 199*199.

(a) Tree

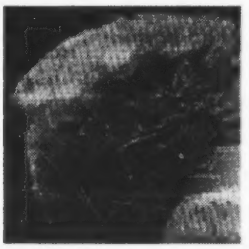


(b) Clock

(c) Stream

Figure 5.1a: Images with high frequencies in the frequency domain.

(d) Mickey



(e) Eagle

(f) Rushmore

Figure 5.1b: Images with low frequencies in the frequency domain.

The tests involve warping each image S to D using a mapping function f and then warping D back to S using a mapping function f^{-1} which is the inverse of f. The original image S is compared with the final warped image for SNR. In our tests, we define two mapping functions as follows:

(1) f_L - warps the image S to twice its size, and
(2) f_S - warps the image S to half its size.

Figure 5.2 shows the mapping functions and their inverses. We implemented an algorithm similar to the Skeleton-based method. For fair comparison of signal-to-noise ratio and speed performance, we chose square image for the tests and image center as the origin for the Radial transform. Therefore both the Skeleton-based and the Radial methods sample the entire image (without calculating background pixel positions) and the areas sampled are the same.

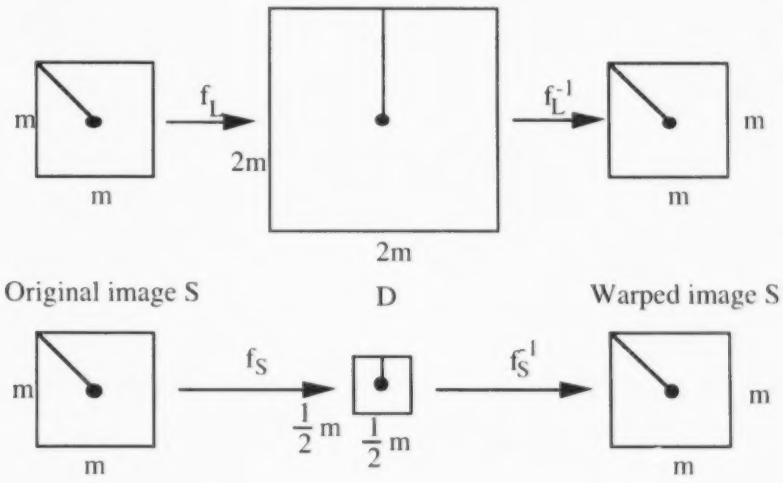


Figure 5.2: Mapping functions used for performance comparisons.

The SNR of these images for the Skeleton-based and Radial methods are illustrated in Figure 5.3. The top row and bottom row of Figure 5.3 are SNR tests for the first and second group of the images, respectively. The left column and right column of Figure 5.3 are transformations using f_L and f_S respectively.

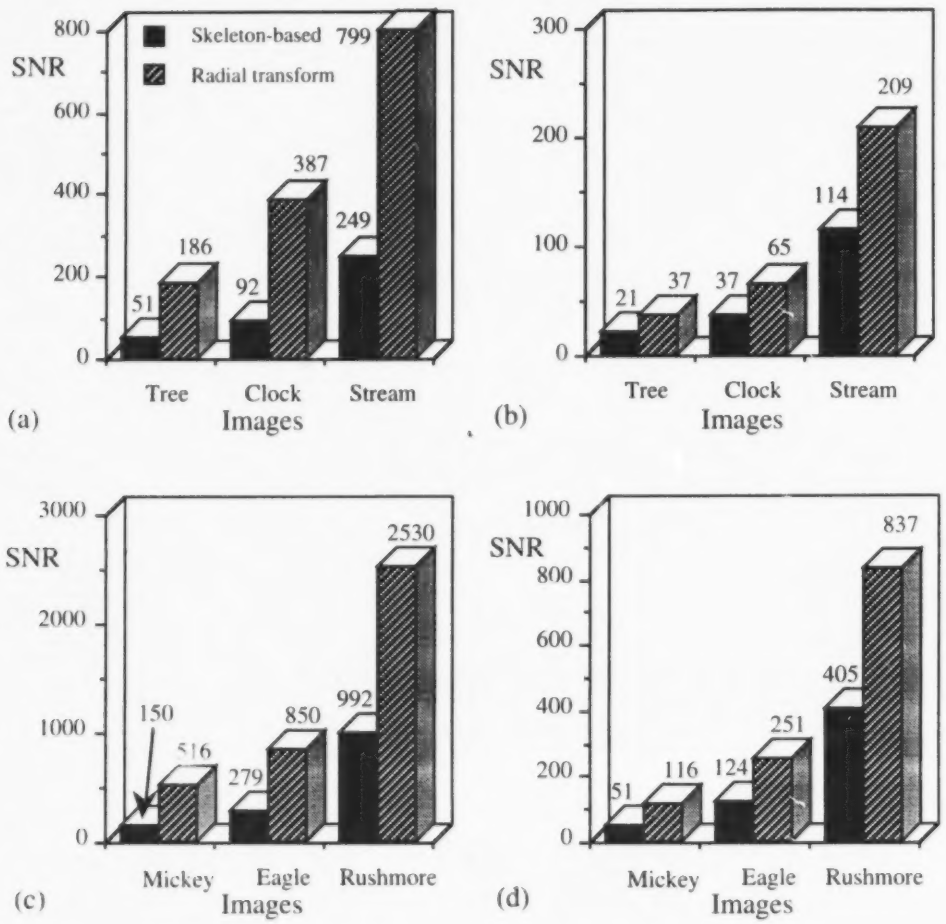


Figure 5.3: Signal-to-noise ratio tests.

The Radial transform outperforms the Skeleton-based technique in all the tests. This is consistent with our theoretical analysis presented earlier. Note the scale difference in the SNR-axis for Figure 5.3. Both Skeleton-based and Radial methods obtain higher SNR using the mapping function f_L than using f_S. The mapping function f_L enlarges the original image and hence intensity information is retained. On the other hand, f_S shrinks the original image and information is lost, therefore SNR drops dramatically. Note that the proportion of drop of SNR in Radial methods is larger than that in the Skeleton-based method. This is reasonable. When using the mapping function f_S, the

destination image D is smaller than the original image S, therefore, D is enlarged first (to be precise the boundary map of D is enlarged). Then S is mapped into D using f_S. But D must be shrunk down again using bilinear interpolation. This is the place where interpolation is "inaccurate". As discussed earlier in the Image Quality Section, both methods interpolate an image radially: the Skeleton-based method interpolates an image along the v-axis and the Radial transform interpolates it along the r-axis. This interpolation is not appropriate when the image is viewed in Cartesian coordinates. Before shrinking D, it has been warped along the r-axis and we start with a deteriorated image. Therefore, the final image quality drops dramatically. For the tests using f_L, D is not shrunk and so this problem does not exist. In any case, the Radial transform still produces a higher quality image than the Skeleton-based method.

We have also warped arbitrary image shapes using both methods. Since these images are difficult to generalize, they are not included. In all those tests we obtained similar SNR results as shown in Figure 5.3.

5.2 Time Comparisons

In the Speed Performance Section we noted that the Skeleton-based method generally transforms an image faster than that of the Radial transform. This is true when the origin in the Radial transform is not at the same skeleton position as in the Skeleton-based method, and if both S and D must be enlarged before the Radial transform radially samples the images. There are a few exceptions where the Radial transform outperforms the Skeleton-based method. For our SNR tests we used square images with the Radial transform origin at the centers of the images, therefore, the origin is at the same position as the skeleton in the Skeleton-based method, and the areas sampled in both methods are the same. All the tests were run on a Sun 4/60C-24 station. Table 5.1 shows the CPU-time range in seconds for the Skeleton-based and the Radial methods for the tests in Figure 5.3. The CPU-time ranges given are for the mapping functions f_L and f_S conducted in the Signal-to-noise Ratio Section. The CPU-time ranges for the inverse of f_L and f_S are similar to that shown in Table 5.1 and therefore are not included.

Table 5.1: CPU-time range.

	Skeleton-based	Radial
f_L	26.8 - 31.2	25.7 - 26.3
f_S	6.7 - 7.2	5.5 - 6.7

The Radial transform outperforms Skeleton-based method in all the tests. It is difficult to give an exact explanation of why Radial transform still performs faster than the Skeleton-based method. We will concentrate on the mapping function f_L. Let the area of image S be $A_1 = m*m$ and the area of image D be $A_2 = 4m*m$. The areas *sampled* for S and D are $A_3 = (4m - 4)(m/2 + 1)$ and $A_4 = (8m - 4)(m + 1)$. The costs of performing all the steps of the Radial transform are given in Table 5.2.

Table 5.2: Cost of performing Radial warping.

(1) cost of scaling S	$Sc(A_2)$
(2) cost of scaling S's boundary map	$Sc(A_2)$
(3) cost of calculating polar coordinates	$2Pc(A_4)$
(4) cost of sampling S to S'	$Sp(A_4)$
(5) cost of mapping S' to D'	$Mp(A_4)$
(6) cost of placing D' onto D	$Sp(A_4)$
(7) cost of scaling D	0
Total costs:	$R = 2Sc(A_2) + 2Pc(A_4) + 2Sp(A_4) + Mp(A_4)$

Assume the time for calculating thinning paths of an image is the same as that for calculating polar coordinates. Also assume the time for sampling an image using the thinning paths is the same as that using theta paths. Let the area of resampling S' along the u-axis in the Skeleton-based method be $A_5 = (8m - 4)(m/2 + 1)$. The costs of performing all the steps of the Skeleton-based method are given in Table 5.3.

Table 5.3: Cost of performing Skeleton-based warping.

(1) cost of calculating thinning paths for S	$Pc(A_3)$
(2) cost of calculating thinning paths for D	$Pc(A_4)$
(3) cost of sampling S to S'	$Sp(A_3)$
(4) cost of resampling S' along the v-axis	$Sp(A_3)$
(5) cost of resampling S' along the u-axis	$Mp(A_5)$
(6) cost of resampling S' along the v-axis	$Mp(A_4)$
(7) cost of placing D' onto D	$Sp(A_4)$
Total costs: $S = Pc(A_3) + Pc(A_4) + 2Sp(A3) + Mp(A_5) + Mp(A_4) + Sp(A_4)$	

The difference of these costs is calculated as follows:

$$S - R = Pc(A_3) + 2Sp(A_3) + Mp(A_5) - 2Sc(A_2) - Pc(A_4) - Sp(A_4) \quad (5.2)$$

From Table 5.1, since $S - R > 0$, then

$$Pc(A_3) + 2Sp(A_3) + Mp(A_5) - 2Sc(A_2) - Pc(A_4) - Sp(A_4) > 0 \quad (5.3)$$

$$Pc(A_3) + 2Sp(A_3) + Mp(A_5) > 2Sc(A_2) + Pc(A_4) + Sp(A_4) \quad (5.4)$$

This is all that we know. Now let us assume that both S and D have the *same* dimension m*m; the total costs for the two methods are then reduced to the following:

$$R = 2Pc(A_3) + 2Sp(A_3) + Mp(A_3) \quad (5.5)$$

and

$$S = 2Pc(A_3) + 3Sp(A_3) + 2Mp(A_3) \quad (5.6)$$

From equations (5.5) and (5.6) we know that the Radial transform is definitely faster than the Skeleton-based method. Indeed, the CPU-time ranges in seconds are shown in Table 5.4:

Table 5.4: CPU-time range for same image size.

	Skeleton-based	Radial
f	10.6 - 10.8	5.4 - 6.0

However, the above case rarely happens with arbitrary images. In all other situations where both S and D must be enlarged for the Radial transform, the Skeleton-based method will outperform the Radial transform.

Chapter 6

Radial Morphing

The Radial transform has an application in morphing which is often used to create visual effects. Morphing, which is derived from "image metamorphosis", refers to image warping followed by cross-dissolve [Wolberg90][Beier and Neely92]. It is often used as a technique for generating special effects. The user is allowed to define key frames for the warp. Both the source and destination images will be warped to the key frames to establish spatial correspondences. Then a cross-dissolve between the two images is performed to interpolate the colors. To make the transformation smooth, in-between frames are added between each key frame. In-between frames are frames automatically created by interpolation. The source image is then gradually warped to the destination image. During the initial sequences of warping, the intermediate images will look more like the source. In contrast, towards the end of the warping sequence the intermediate images will look more like the destination. The sequences are usually displayed rapidly to generate the visual effects. It is also possible to specify the rate of cross-dissolve so that part of the image colors may be changed quicker than the others for additional effects.

The Radial transform can be used to morph images of arbitrary shapes. Consider interpolating Line 1 to Line 2 in Figure 6.1. For linear interpolation, we join the end-points of the lines by dotted lines. The interpolated line (or the in-between line) lies halfway between the dotted lines. To achieve smooth motion, we can split the dotted lines as many segments as needed. All we have to do is divide the length of each dotted line by the number of interpolations.

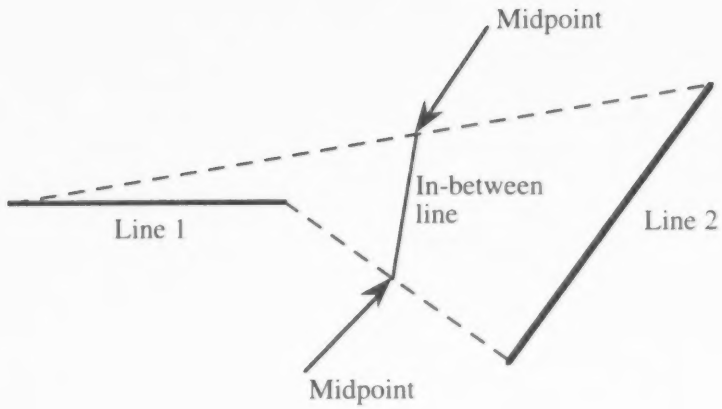


Figure 6.1: Linear interpolation of two lines.

The interpolation of the radial lines for the Radial transform is similar to the above example. As before, the bounding boxes of the source and destination images must have the same size. If this is not the case, the smaller images are enlarged to match the largest dimensions. For each radial line in the source image and its correspondence in the destination, find the positions of the in-between lines as described above. The in-between lines are used to warp the source image into a sequence of in-between frames. Each in-between frame is warped using the Radial transform. The rotational direction for each frame must be the same within this sequence. Repeat the warping process for the destination image in the *reverse* direction. We now have two sequences of in-between frames. Cross-dissolve each in-between frame from the source and destination image sequences. We then have a sequence of morphed frames. These frames all have the same dimension. If the original source and destination images different in size, then the morphed frames must be downscaled. We could use linear interpolation again to determine the size of each frame. We are now ready to display the morphed image-sequences.

The above process morph images of arbitrary shapes. If the position of the origins of the source and destination images is different, the morph will shift the origins also. The only restriction of this algorithm is that both original images should not contain

holes or islands. The introduction of holes or islands in the images may not ensure a smooth transition between the morphed frames.

Chapter 7

Conclusion

This thesis presents a simple Radial transform that warps images which are delimited by any closed planar curves. The polar coordinate representation of arbitrary image shapes provides a closer description of the relationship between image boundaries and interior points. The Radial transform also provides more control over the warping process by allowing origins and sampling directions to be defined in the images. When both the source and destination images are scaled to the same dimension, the correspondence problem, which is the most difficult issue in this type of warping algorithms, is solved almost trivially. The sampling directions allows either rotation or rotation with lateral transformation. The bounding-box traversal of the source image eliminates "holes" by redundantly sample the image points. We have shown exactly that this redundancy is bounded by an upper limit which is a function of the maximum image dimension. The boundary maps not only provide a convenient method to sample the foreground pixels, but also facilitate the warping of non-topological equivalent images. However, this kind of warping may not necessarily describe a smooth transition between the breaking points of radial-line segments.

This thesis also presents a Radial-axes algorithm which is the combination of the Radial spatial transformation and the Skeleton-based 3-pass transformation. The Radial-axes algorithm provides even more control over the warping process than that of Radial transform.

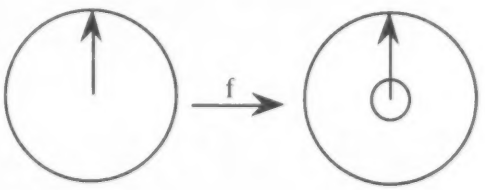
Although the Radial transform produces a high quality image; simplifies the correspondence problem, and hence the resampling process; and allows the mapping of non-topological equivalent images, there are limitations. The arbitrary source image may not have an "even" stretch over its entire area. Another desirable feature is to smooth out the breaking points of line segments for non-topological equivalent warping. In terms of complexity of the algorithm, it is also desirable to reduce the time required for transforming images. One last disadvantage of the Radial transform is the requirements of large memory space. With presently available inexpensive memory hardware, this is not a major problem.

Chapter 8

Gallery

This section provides several examples of the transformation of some images. The example images of the Radial transform are given first, followed by that of the Radial-Axes transform. The images on the left and right columns are the source and destination images respectively. Their corresponding origins are also given.

Radial transform: the rotational directions for all images are the same.



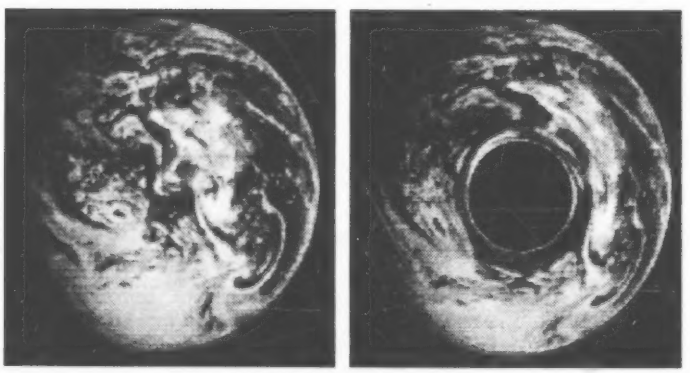
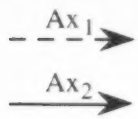
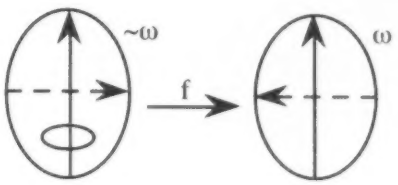


Figure 8.1: Ozone

Radial-Axes Transform: the rotational directions of some images are different.





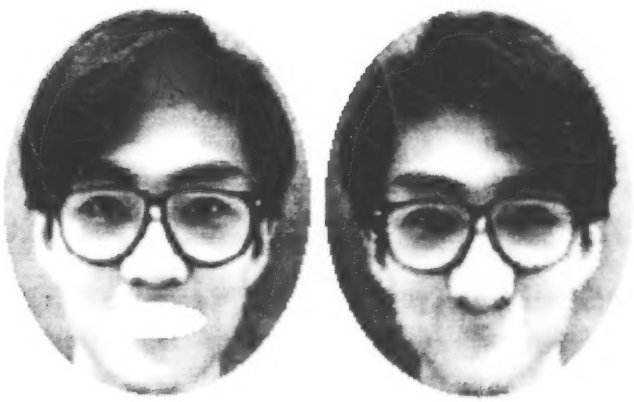


Figure 8.6: Silence
(lateral transform on non-topologically equivalent images)

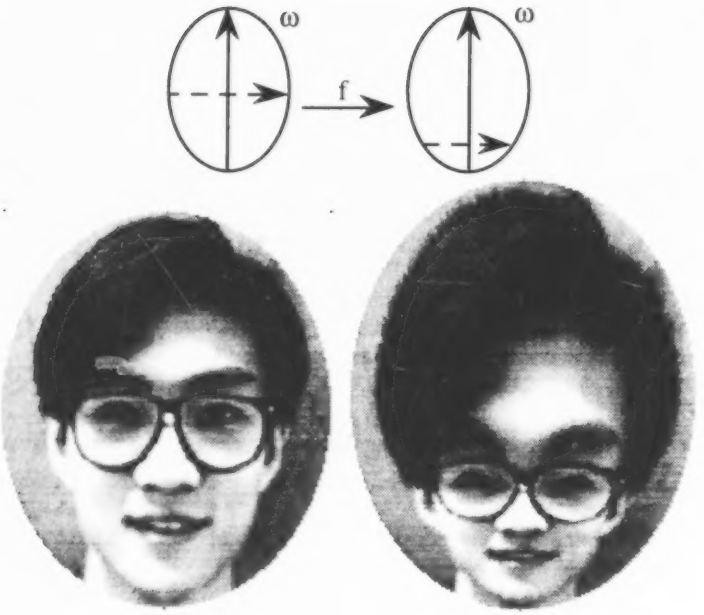


Figure 8.7: Alien

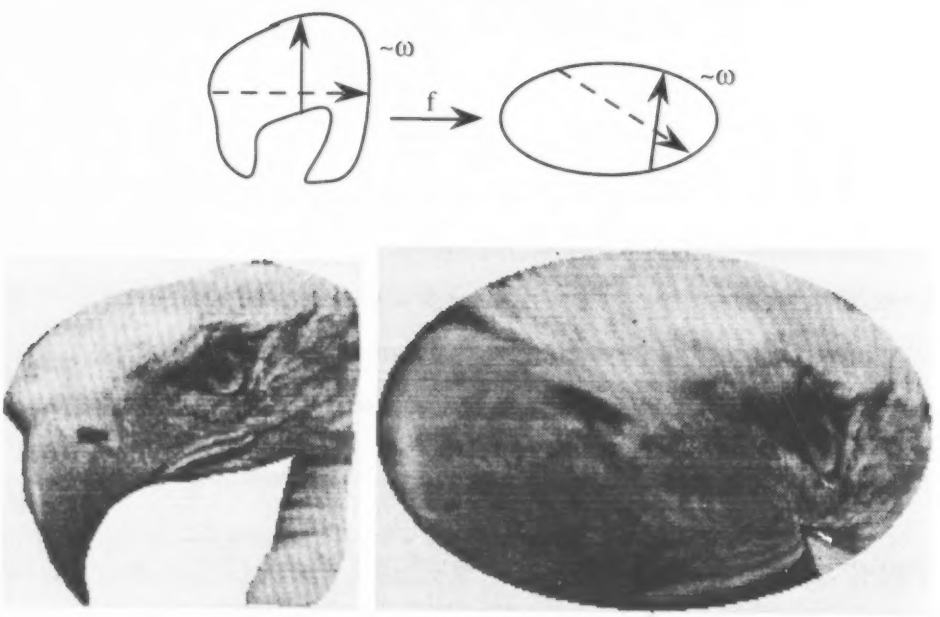


Figure 8.8: Eggle

Bibliography

[Anderson93] S. Anderson, "Morphing Magic", *SAMS*, Carmel, IN, 1993.

[Beier and Neely92] T. Beier and S. Neely, "Feature-Based Image Metamorphosis", *Computer Graphics* (SIGGRAPH '92), Vol. 26, No. 2, July, 1992, pp. 35-42.

[Braccini and Marino79] C. Braccini and G. Marino, "Fast Geometrical Manipulations of Digital Images", *Computer Graphics and Image Processing*, Vol. 13, 1979, pp. 127-141.

[Breene and Bryant93] L.A. Breene and J. Bryant, "Image Warping by Scanline Operations", *Comput. & Graphics* Vol. 17, No. 2, 1993, pp.127-130.

[Burt et al.83] P.J. Burt, C. Yen, and X. Xu, "Multi-resolution Flow-through Motion Analysis", *Proc. IEEE CVPR Conf.*, 1983, pp. 246-252.

[Butler89] T. Butler, "Three Approaches to Terrain Rendering", *Proc. SPIE*, Vol. 1075, Jan., 1989, pp. 217-225.

[Catmull and Smith80] E. Catmull and A.R. Smith, "3-D Transformations of Images in Scanline Order", *Computer Graphics* (SIGGRAPH '80 Proc.), Vol. 14, No. 3, July, 1980, pp. 279-285.

[DiPaola91] S. DiPaola, "Extending the Range of Facial Types", *Journal of Visualization and Computer Animation*, Vol. 2, 1991, pp. 129-131.

[Fant86] K.M. Fant, "A Non-Aliasing, Real-Time Spatial Transform Technique", *IEEE Computer Graphics & Applications*, Jan., 1986, pp. 71-80.

[Fiume et al.87] E. Fiume, A. Fournier and V. Canale, "Conformal Texture Mapping", *Proc. Eurographics*, 1987, pp. 53-64.

[Foley et al.90] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, "Computer Graphics -- Principles and Practice", *Addison-Wesley*, 1990.

[Heckbert86] P.S. Heckbert, "Survey of Texture Mapping", *Proc. Graphics Interface*, 1986, pp. 56-67.

[Holzmann88] G.J. Holzmann, "Beyond Photography -- The Digital Darkroom," *Prentice-Hall*, Englewood, Cliffs, NJ, 1988.

[Huang and Hsu81] T.S. Huang and Y.P. Hsu, "Image sequence enhancement", In T.S. Huang, ed., *Image Sequence Analysis*, Springer-Verlag, Berlin, 1981, pp. 290-310.

[Magnenat-Thalmann at el.88] N. Magnenat-Thalmann, H. T. Minh, M. de Angelis, and D. Thalmann, "Human Prototyping", *New Trends in Computer Graphics* (Proc. Computer Graphics Intl. '88), Ed by N. Magnenat-Thalmann and D. Thalmann, Springer-Verlag, 1988, pp. 74-82.

[Magnenat-Thalmann at el.89] N. Magnenat-Thalmann, H. T. Minh, M. de Angelis, and D. Thalmann, "Design, Transformation and Animation of Human Faces",*Visual Computer*, Vol. 5, 1989, pp. 32-39.

[Morrison93] M. Morrison, "The Magic of Image Processing", *SAMS*, Carmel, IN, 1993.

[Nahas et al.90] M. Nahas, H. Huitric, M. Rioux, and J. Domey, "Facial Image Synthesis Using Skin Texture Recording", *Visual Computer*, Vol. 6, 1990, pp. 337-343.

[Quam84] L.H. Quam, "Hierarchical Warp Stereo", In *Proc. DARPA Image Understanding Workshop*, 1984, pp. 149 - 156.

[Singh90] A. Singh, "An Estimation-Theoretic Framework for Image Flow Computation", In *Proc. Third Int'l Conf. Computer Vision*, Osaka, Japan, 1990, pp. 168 - 177.

[Singh91] A. Singh, "Optic Flow Computation — A Unified Perspective", *IEEE Computer Society Press*, 1991.

[Smith87] A.R. Smith, "Planar 2-pass Texture Mapping and Warping", *Computer Graphics* (SIGGRAPH '87 Proc.), Vol. 21, No. 4, July, 1987, pp. 263-272.

[Trainer and Sun91] T.J. Trainer and F.K. Sun, "Image Resampling in Remote Sensing and Image Visualization Applications", *Proc. SPIE*, Vol. 1567, Applications of Digital Image Processing XIV, 1991, pp. 650-658.

[Walterman and Weinhaus91] M. Walterman and G. Weinhaus, "Antialiasing Warped Imagery using Look-up Table Based Methods for Adaptive Resampling", *Proc. SPIE*, Vol. 1567, Applications of Digital Image Processing XIV ,1991, pp. 204-214.

[Ward and Cok89] J. Ward and D.R. Cok, "Resampling Algorithms for Image Resizing and Rotation", *Proc. SPIE*, Vol. 1075, Digital Image Processing Applications, 1989, pp. 260-269.

[Weinhaus and Walterman90] G. Weinhaus and M. Walterman, "A Flexible Approach to Image Warping", *Proc. SPIE*, Vol. 1244, Image Processing Algorithms and Techniques, 1990, pp. 108-122.

[Wolberg and Boult89] G. Wolberg and T.E. Boult, "Separable Image Warping with Spatial Lookup Tables", *Computer Graphics* (SIGGRAPH '89 Proc.), Vol. 23, No. 3, July, 1989, pp. 369-378.

[Wolberg88] G. Wolberg, "Image Warping Among Arbitrary Planar Shapes", *New Trends in Computer Graphics* (Proc. Computer Graphics Intl. '88), Ed. by N. Magnenat-Thalmann and D. Thalmann, Springer-Verlag, 1988, pp. 209-218.

[Wolberg89] G. Wolberg, "Skeleton-Based Image Warping", *Visual Computer*, Vol. 5, 1989, pp. 95-108.

[Wolberg90] G. Wolberg, "Digital Image Warping", *IEEE Computer Society Press*, 1990.

[Yau and Duffy88] J.F. Yau and N. D. Duffy, "3-D Facial Animation Using Image Samples", *New Trends in Computer Graphics* (Proc. Computer Graphics Intl. '88), Ed by N. Magnenat-Thalmann and D. Thalmann, Springer-Verlag, 1988, pp. 64 - 73.